Department of Mechanical Engineering
University of Canterbury
Te Whare Wānganga o Waitaha
Private Bag 4800
Christchurch 8020, New Zealand

Telephone: +64-3-366 7001
Facsimile: +64-3-364 2078
Website: www.mech.canterbury.ac.nz

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

University of Canterbury

Masters Thesis

Development of an autonomous surface tracking quadrotor system utilizing low-cost proximity sensors

Masters Student: Jamie Spyker
Supervisor: XiaoQi Chen

August 28, 2017

Abstract

This thesis provides extensive research and development for autonomous control of a quadrotor system using low-cost proximity sensors. Investigations towards this are completed in the areas of quadrotor simulation, control algorithm development and prototype system construction.

Autonomous UAV systems can be highly useful for a variety of applications and are becoming more widespread in modern society. Reviewing current techniques for autonomous control showed that most commonly computer vision and GPS methods are utilized, with their key downsides being high complexity and cost. It was proposed that creating a simplified autonomous control system using proximity sensors could be option for reducing these problems.

The use for such a system was identified to have a potential application within the areas of commercial inspections, where a nearby surface would constantly be present for ranging data. A literature review was completed into current similar applications and methods providing a basis to build off. Further background research was completed into UAV dynamics and relevant hardware for proximity sensor tracking.

To enable the development of algorithms with use of proximity sensors, simulation software was created. The base simulation implementation included dynamic modelling of a quadrotor and localized control methods. Following this virtual sensors and surfaces were designed and implemented into the simulation software. The developed software was functionally tested and worked as expected comparing well to results of similar research.

With use of the built simulator, research was undertaken into potential control algorithms utilizing proximity sensors. The intention of this exploration is to confirm the viability of proximity sensor control and to provides potential algorithm options that can be extended upon in the future. Control algorithms were developed which focused on automating a singular control channels - for roll, pitch and yaw of a quadrotor system. Options were presented for both flat and varied vertical surfaces proving the potential viability for such methods.

Development for a prototype system was completed that can control a UAV with autonomous isolated channels, for the purpose of real-world testing of algorithms. The system was validated by testing a prior control algorithm created within the simulator. The results of the functional tests showed the the system successfully tracking a vertical surface with autonomous pitch control. Further investigation into the performance results showed that improvements should be focused on reducing system noise as significant fluctuations were present reducing control accuracy.

In summary, this study provides a comprehensive overview of the potential options for autonomous control of a quadrotor using proximity sensors and proves their viability. Key contributions include the addition of virtual sensors in a simulation environment, algorithm options for autonomous control and the development of autonomous control hardware. This investigation provides a solid basis for future researchers with wide possibilities of potential future work that could be completed.

Contents

1 Introduction **8**
- 1.1 Motivation . . . 8
  - 1.1.1 UAV Background . . . 8
  - 1.1.2 UAV Application within Commercial Inspections . . . 8
  - 1.1.3 Current Techniques towards Autonomous Control . . . 11
  - 1.1.4 Possibility of Proximity Sensor Control . . . 12
- 1.2 Problem Domain . . . 13
- 1.3 Contribution of Work . . . 14
- 1.4 Thesis Overview . . . 14

2 Background and Literature **15**
- 2.1 Background of Hardware . . . 15
  - 2.1.1 Quadcopter . . . 15
  - 2.1.2 Ultrasonic Sensors . . . 16
  - 2.1.3 Laser Range Finder . . . 17
  - 2.1.4 IMU . . . 17
- 2.2 Review of Wall Following Techniques . . . 18
- 2.3 Review of Simulation Methods . . . 20
- 2.4 Review of UAV Control Utilizing Proximity Sensors . . . 22
- 2.5 Summary . . . 24

3 Simulation of quadrotor system **25**
- 3.1 Requirements . . . 25
- 3.2 Simulation Setup . . . 25
  - 3.2.1 Software . . . 25
  - 3.2.2 Fixed Constants . . . 25
  - 3.2.3 Overview . . . 26
- 3.3 Dynamic Model . . . 26
- 3.4 Surface Modelling . . . 29
  - 3.4.1 Implementation . . . 29
- 3.5 Sensor Modeling . . . 30
  - 3.5.1 Initial Placement . . . 30
  - 3.5.2 Intersection Algorithm . . . 31
  - 3.5.3 Full Intersection Search . . . 36
  - 3.5.4 Local Region Intersection . . . 38
- 3.6 UAV Controller . . . 38
  - 3.6.1 UAV Stabilization and Control . . . 38
- 3.7 Functional Testing of Simulation . . . 41
  - 3.7.1 Dynamic Model and Control Validation . . . 41
  - 3.7.2 Model Surface and Sensor Validation . . . 42
  - 3.7.3 Performance . . . 43
- 3.8 Summary . . . 45

4 Development towards UAV control methods using proximity sensors **46**
4.1 Initial Simulated Hardware 46
4.2 Hovering Algorithm 46
4.3 Flat Surface Algorithm Development 47
4.3.1 Sensor Placement 47
4.3.2 Vertical Surface Alignment through Yaw Control 52
4.3.3 Maintaining Vertical Surface Distance through Pitch Control 53
4.3.4 Parallel Movement Alongside Vertical Surface with Roll Control 57
4.4 Profiled Surface Algorithm Development 58
4.4.1 Sensor Placement 58
4.4.2 Vertical Surface Alignment through Yaw Control 60
4.4.3 Maintaining vertical Surface Distances through Pitch Control 61
4.4.4 Parallel Movement alongside Vertical Surface through Roll Control 64
4.5 Summary 65

5 System Design and Validation **67**
5.1 Requirements 67
5.2 Hardware 68
5.2.1 Flight Platform 68
5.2.2 Modifications 68
5.3 Electronics 69
5.3.1 On-board Microcontroller 69
5.3.2 Sensor Selection 70
5.3.3 Data Logger 71
5.3.4 Modular Shield 71
5.3.5 Additional Operation Mode Microcontroller 73
5.4 Complete System 73
5.5 Software 74
5.5.1 Main 75
5.5.2 Sensors 76
5.5.3 Data logging 76
5.5.4 Radio Emulation 76
5.5.5 Motor Control 77
5.5.6 Secondary Microcontroller - Operation Mode 77
5.6 Testing Results 77
5.6.1 Testing Environment 77
5.6.2 Isolated Pitch Control Experimental Testing 77
5.7 Summary 81

6 Conclusion **82**
6.1 Overview and Contribution Summary 82
6.2 Viability for practical use 82
6.3 Further Expansion and Future Work 83
6.3.1 Simulation Accuracy 83
6.3.2 Simulation Optimizations 84

6.3.3 Hybrid Control 84
6.3.4 Autonomous Control for Specific Situations using Proximity Sensors 85
6.3.5 Integration with proven methods 85
6.3.6 Prototype Development 85
6.4 Closing Remarks 86

7 Bibliography **87**

8 Appendices **94**

List of Figures

1 Examples of different types of UAVS. 9
2 Example of a Quadrotor 15
3 How control is achieved on a Quadrotor 16
4 Singular proximity sensor wall following behaviour 19
5 Autonomous robot detects and avoids front-left surface was using a sonar-ring 19
6 Visualization of the blade flapping phenomenon 21
7 Visualisation of proximity sensors providing distances to enclosed room surfaces 23
8 UAV Free body diagram 27
9 Modelling surface plane structure 29
10 Example of a modelled surface plane 30
11 Rotation of 2 offset sensors along y-axis in body frame 32
12 Barycentric Coordinates 34
13 Plane intersections from four possible 3-point coordinate combinations 37
14 Double Intersection with surface model 38
15 Visual representation of local region search 39
16 Simulation of quadrotor traveling between waypoints 42
17 Simulation of a localized quadrotor moving horizontally at a fixed speed 43
18 Comparison of measured sensor data to a profiled surface 44
19 Height of quadcopter over time using hovering controller 48
20 Parallel quadcopter system with two sensors ranging a vertical surface 49
21 Angled quadcopter system with two sensors ranging a vertical surface 49
22 Angled quadcopter system with 2 locally spaced apart sensors ranging a vertical surface 50
23 Visual Example of Specular Reflection 52
24 UAV in parallel sensor configuration 53
25 Adjusting yaw over time tracking a flat surface 54
26 Birds eye view of simulated UAV adjusting yaw over time 54
27 View from onboard sensors of quadcopter over time 55
28 Adjusting pitch over time tracking a flat surface 56
29 View of simulated UAV adjusting pitch over time 56
30 Birds eye view of simulated UAV adjusting roll over time. 57
31 Angled sensor moving parallel to profiled vertical surface of time 59
32 Angled sensor measurements moving parallel to a profiled surface over time. 59
33 UAV in hybrid sensor configuration. 60
34 Birds eye view of simulated UAV adjusting yaw over time 61
35 Simulation of UAV tracking a profiled vertical surface using a lookahead algorithm 62
36 Simulation of UAV tracking a inside corner surface with roll adjustment control 65

37 Simulation of UAV tracking a inside corner surface without roll adjustment control 66
38 Solidworks Model of Quadcopter 69
39 Quadcopter prototype system 70
40 Simplified Diagram of Buffer Plan for controlling Quadcopter . . 72
41 Prototype PCB Shield 73
42 Quadcopter System Architecture 74
43 Simplified Diagram of Buffer Plan for controlling Quadcopter . . 75
44 Testing Environment for Prototype Quadrotor System 78
45 Tracking results of experimental testing 79
46 Output control gains of experimental testing 79
47 Comparison of derivate control and sensor measurements from experimental testing 80

List of Tables

1 Fixed UAV Constants 26
2 Variable Simulation Constants 41
3 Navigation Waypoints 42
4 Parameters used to generate random varied surface 43
5 Summarized Simulation Performance Results 44

1 Introduction

This chapter details the use of UAV systems and their potential for autonomous control. Motivations are presented in regards to the application of using an UAV system for commercial inspections. The current limitations towards this specific application are explored to determine what improvements could be made. Current general techniques for autonomous control are also reviewed. The possibility of autonomous control using basic sensors is then discussed, alongside the scope of the contributions that would be made.

1.1 Motivation

1.1.1 UAV Background

An Unmanned Aerial Vehicle (UAV) is a form of aircraft without an on-board human pilot. There are many different type of UAV systems available which are used for different purposes, with some examples being seen in Figure 1. In today's society UAVs are becoming more widespread for many different applications such as military usage, aerial imaging and commercial inspections [1–3]. They are also are receiving a sizeable amount of attention in current academic fields [1]. In particular, this thesis is interested in MAVs (Micro Aerial Vehicles). These are a class of UAVs that are size restricted, allowing them to be suitable for a wide range of tasks which larger vehicles could not achieve or are superfluous for [4, 5].

Any UAV requires some form of control to operate the systems navigation as required. Traditionally this is done by a human operator, but advances have been made to allow for these UAVs to operate autonomously with use of autopilot systems [7–9]. These are useful methods of operation, but they do not cover all possibilities of desired autonomous control and there are still many problems that need to be addressed.

Due to the low cost and small size of MAVs they can be easily specialized to a particular task. When creating a specialized UAV system the development of the research, hardware and software would account for a large portion of the cost and thus once completed it would be inexpensive to mass the product itself. This indicates that developing a MAV system for widespread use in a particular field would be highly beneficial. Furthermore, ideally the designed system would have minimal complexity which infers that the development costs would also be lower; thus reducing a significant portion of the cost.

1.1.2 UAV Application within Commercial Inspections

One application of UAVs is using them for commercial inspections in areas that are otherwise difficult to assess, such as tall vertical structures or tight enclosed

[1] Engineering bibliographic database Compendex shows over 13500 relevant results in the past 5 years alone (2013-2017)

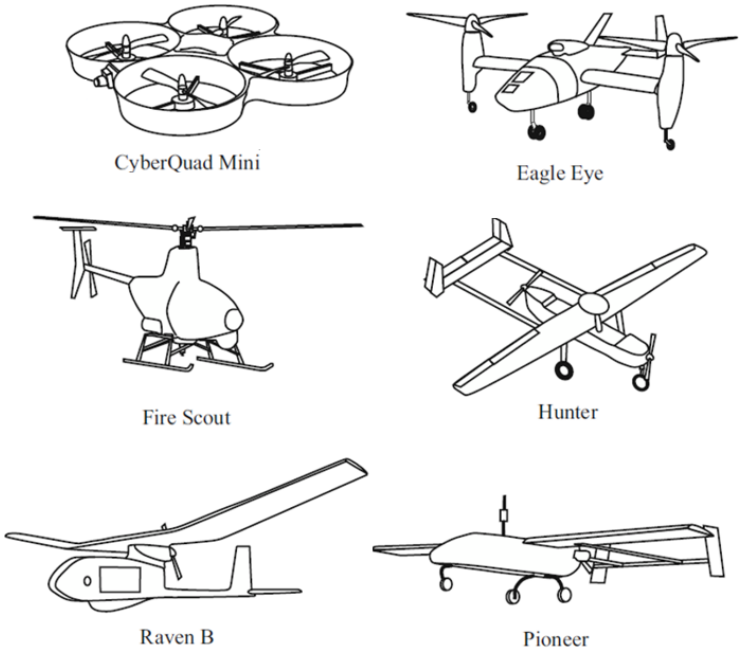


Figure 1: Examples of different types of UAVS. [6]

spaces [10]. Utilizing UAVs for inspection purposes is useful as it solves problems present in former methods used throughout history. In the past inspections have been made through climbing and rappelling structures, which can be slow and hazardous. Other options included using aerial work platforms in which a human inspector would be placed on to look at some structure more closely. These are limited in the height that they can extend as well as where the base vehicle can be maneuvered to [11].

Currently commercial inspections can be completed using UAVs, but these require using a highly skilled operator to control the system manually. Completely automating this process would be highly useful for multiple industries, saving both time and money. This a highly technical challenge, but even making progress towards this goal could be useful for other means. Hybrid control could be utilized to prevent operators from accidentally crashing into close proximity surfaces, or allow operators to maintain a fixed steady distance from a vertical surface. Whatever the precise application, there are many specific challenge that must be overcome to make progress towards a complete autonomous system.

Computational Demands

One major limitation present for any application of UAVs is the computation limits of the on-board computer. A recent general survey of infrastructure methods identified this as one of the key challenges for UAV vision and control methods due to the high computational demand compared to the processing

capabilities. [12]. Depending on the UAV in question there also may be weight and cost limits for the on-board computer which means that there is limits to the computation that can be achieved. Many advanced techniques of navigation can be very computationally expensive, as they make use of advanced control and computer vision methods.

However as technology is rapidly advancing this factor is becoming less and less of an issue, as computation power will be abundant. Even so, it is prudent to explore alternate strategies of reducing computational load by using optimized algorithms or simpler navigation methods.

GPS-Denied Enviroment

Many inspection tasks are situated in areas in which GPS may not be available. This entails that the UAV in question must be able to be completely self-sufficient in absence of GPS for the system to be viable in all applications of commercial inspection.

Battery Life

The operation time of a UAV is entirely dependent on the life of the onboard battery. Depending on the type of battery and what your UAV weight this can vary, but the limitation will always be there that it is operating on a restricted time period. For commercial inspection purposes, ideally the battery life should be long enough to fully inspect a entire desired structure without having to return to base. This would varying depending on the exact application of the system.

Safety

Commercial inspections imply operating in close range to objects or surfaces. When operating in close proximity to vertical surfaces it is important that the system has highly accurate localization such that it does not drift and crash into any of the nearby obstacles. First and fore mostly this is due to the aspect of safety. If the UAV were to crash it could endanger nearby personnel or come to rest in a dangerous location to retrieve. Secondarily, a system that crashes frequently can damage the hardware and incur recurring costs that could be otherwise avoided.

Video Stability

When inspecting a commercial structures, on-board video cameras are commonly used. Maintaining the stability of the UAV is highly important for obtaining video footage of a subject in question. Shaky and unstable systems would not provide clear footage making it more difficult to process and inspect at a later date. Aerial vehicles have the potential to move at very high speeds, meaning that motion blur could affect the desired footage in some scenarios.

Even if full stability is achieved, another factor to consider is the movement of quadcopter itself. To move different directions quadcopters pitch, roll and

yaw which change the angle of the quadcopter. These fast bank angle changes need to be accounted when designing a system so that the effect is reduced [10].

1.1.3 Current Techniques towards Autonomous Control

GPS

The most straight forward method of navigation is to use GPS to navigate some UAV where required. With advances in modern technology these GPS units can be quite affordable and reasonably accurate; but still can run into issues for certain applications. When operating in close proximity to obstacles accuracy is highly important and cannot be prone to any error such that no collisions can occur. One such implementation in prior work showed a full positional accuracy of less than 1.4m which is sufficient for general exploration but not for precise environments in which a UAV needs to operate in close proximity to obstacles [13]. Another drawback is that relying too heavily on a GPS system also limits the system as it cannot be run in GPS denied areas. This may not be an issue for UAV projects which are focused on areas where a strong GPS signal is present, but must be considered if the system needs to work on all possible locations.

Computer Vision

Computer vision is another valuable option for the autonomous navigation of UAV systems and there is a multitude of different techniques available that can be used [14]. The major problem many computer vision algorithms encounter however is that they can be very prone to disturbances such as changes in lighting conditions or unplanned motion from environmental effects, which could be often present in UAV systems [15]. Many of the algorithms created can also be very computationally intensive and unsuitable for small UAV systems. Listing all techniques possible would be extraneous, so only a few common methods for UAV control will be explored here.

One computer vision technique used for the control and navigation of UAVs is optical flow. This method measures the apparent motion of some video feed caused by the relative motion between the camera lens and the scene. This can then be used to imply data about the UAV such as the current position or speed. Solely using this form of navigation is not perfect however and position error can slowly increase over time due to accumulated system, sensor and control noise. [16].

Feature tracking is another method that can be used for autonomous control. Image processing techniques can be utilized to find features in an image that can be used to identify any objects of interest. Examples of some processing techniques that can be used are points or edge extraction, but these are not always necessary. These key features can then be tracked between frames to and provide a basis for flight navigation and control. When using this technique desired characteristics of the input footage is smooth motion, minimal occlusion, illumination consistency and distinct objects [17]. These characteristics can be

difficult to achieve on a UAV system.

Stereo vision is another method that can be used to help navigate a UAV system. Stereo vision works by using two calibrated cameras together to track certain points of interest. Matching these points between the two cameras the relative depth of the point can be obtained, forming a disparity map. This could be useful for height or motion estimation of the UAV.

Each of these methods presented can be valuable in specific circumstances and adapted to a various tasks as necessary, leading computer vision methods to be a versatile option for autonomous control. However, the limitations should always be reviewed and considered.

SLAM

The Simultaneous Localisation and Mapping Algorithm (SLAM) algorithm addresses the problem of having a system formulate and update a map of an unknown environment, whilst simultaneously tracking its location within the map. In the right environment, this would allow a UAV to fly around some area and generate a map while continually knowing it's own position within that space. This algorithm is implemented using additional sensors, such vision methods, LIDAR or similar [18, 19].

The SLAM algorithm generally consists of multiple stages: Landmark extraction, data association, state estimation, state update and landmark update. As the UAV moves around some environment, landmarks are extracted relative to the system's position with the use of sensors. Following this, the robot attempts to associate these landmarks with prior observations of its environment. Landmarks that have been previously observed are then used to update the UAV's position within the environment. New landmarks that have just been encountered for the first time are stored as new observations so they can be compared against later observations [20].

The SLAM algorithm comes with it's own trade-offs to consider. These include the requiring large amount of computational power for processing state estimation and the issue of increased complexity for larger maps. [21]. One of the goals for this project is to minimize the computational power and keep the system simple, and so this technique will not be utilized.

1.1.4 Possibility of Proximity Sensor Control

A more underutilized approach is to use proximity sensors as feedback to enable autonomous control. This form of sensors can return distance measurements to objects in the world to help inform the current position of the UAV system. This could be used for real-time adjustments of a UAV system or to assist in localizing the position of the device. There are many potential options to choose from when selecting sensors and each of these have different trade-offs to consider. Such options include ultrasonic sensors, laser-range finders, LIDAR and infrared sensors.

This area of autonomous navigation has been left relatively unexplored compared to the popular GPS and computer vision techniques. Searching the engineering bibliographic database Compendex with the parameter "UAV" returned 1500+ results in conjunction with parameters "GPS" or "Computer Vision". In comparison, less than 100 results were found for parameters "Proximity Sensor", "Laser Range Finder" or "Ultrasonic Sensors" when searched in conjunction with the "UAV" parameter (July 2017).

In relation to commercial inspections this could be a highly useful area to explore further, as when inspecting objects there would always be a surface present to provide feedback to the UAV through these sensors. Development of a low-cost autonomous solution could be highly beneficial for industries that rely on such inspections. Commercial inspections are one particular use case, and the system could be extended to use for any application that has nearby surfaces present. There are many further benefits of exploring this method in particular for such a purpose, which include but are not limited to the following:

- Low Complexity
- Works in GPS-Denied Areas
- Robust to changes in lighting conditions
- Accurate control in close proximity to hazardous surfaces.

1.2 Problem Domain

This project focuses on providing research and development towards achieving autonomous control of a UAV system relying predominately on proximity sensors as a means of feedback. It is desired for the project to take a multi-faceted approach to provide initial research within the areas of simulation, algorithm development and prototype creation. This proposed scope allows for a wide range of groundwork to be completed, which can be then used as a basis for future work in similar areas.

The goal of the simulation to create a platform to allow for the testing of various algorithms with use of proximity sensors. The focus will remain toward virtual sensor integration development within the simulation alongside proven dynamic models. The dynamic model used will focus on modelling of the core UAV systems sufficient enough to provide valuable feedback for experimental algorithms. The built simulation is not intended to be a highly accurate model detailing all the possible intricacies of complex UAV system, but instead to provide a sufficient test environment to experiment with control algorithms.

Unconstrained algorithm development can cover a wide area, even focusing solely on proximity sensors. To alleviate this, the scope is reduced to isolated control inputs for both flat and varied surfaces. This enables a variety of unique algorithms and methods to be tested in a controlled environment, which can then be built upon on in future work.

The scope of the prototype builds off the algorithm development goals, in providing a platform to test algorithms using isolated control inputs. This simplifies testing procedures as one degree of freedom can be monitored, controlled and optimized without interference.

1.3 Contribution of Work

The particular contributions of this work will be in the following areas.

Dynamic modelling of UAV system with focus on sensor integration

A simulation environment will be created to model the dynamics of a UAV system, based off prior work. Virtual sensors and surfaces will be integrated into this environment using a novel approach. This will enable experimentation of algorithms which utilize proximity sensors within the simulation.

Investigation of real-time proximity control algorithms

Investigations will be completed into possible control algorithms utilizing proximity sensors for isolated control inputs. These will consider both basic flat surfaces and surfaces of varying profiles. Considerations and development towards these methods will both be presented.

Development of autonomous control hardware for UAV systems

Control hardware and software will be designed to autonomously control a UAV. This system will remain modular and transferable between differing UAV systems, and have the capacity for isolating control inputs. A prototype UAV will be constructed to confirm the functionality of the overall system and undergo basic tests of autonomous control using proximity sensors.

1.4 Thesis Overview

This thesis describes the research and development towards a quadrotor capable of autonomous flight using proximity sensors. Chapter 2 first provides an overview of quadrotor and sensor hardware, followed by a review of literature with a focus on autonomous wall following, simulation methods and control methods using proximity sensors. Chapter 3 presents the development of a quadrotor simulator with the addition of virtual proximity sensors. Chapter 4 investigates various potential algorithms for autonomous quadrotor control using proximity sensors. Chapter 5 presents the development of a modular quadrotor platform with the capability of isolating control channels. This includes hardware and software design, followed by validation of the system and experimental testing. Chapter 6 concludes the thesis by summarizing the key findings and recommends options for future work.

2 Background and Literature

This chapter outlines further background research completed and details the literature review undertaken towards this project. Key pieces of hardware were reviewed examining their respective benefits that they bring along with their disadvantages. A variety of current autonomous navigation methods were explored to ensure a correct approach was being taken. A review of the current literature was completed, detailing the current state of the art research towards UAV navigation. In particular a focus was placed on wall following, simulation and proximity sensor navigation.

2.1 Background of Hardware

2.1.1 Quadcopter

Many different types of UAVs are available in today's market, but this thesis will focus on quadcopters. An example of a quadcopter can be seen in Figure 2. This form of UAV was selected are they are mechanically simple, easy to control, small, highly maneuverable and have sufficient endurance for inspection missions [22]. There are many off the shelf products available when buying quadcopters, allowing the project time and resources to focus on other areas of research using the quadcopter as a base platform.

Figure 2: Example of a Quadrotor

Quadcopters are a multirotor type of helicopter which utilize 4 separate rotors to lift and move the device. Control is achieved through independent control of each of these rotors allowing the quadcopter to pitch, yaw, roll and change altitude with ease. Figure 3 shows how this can be attained by rotating each of the rotors in specific directions with a corresponding torque.

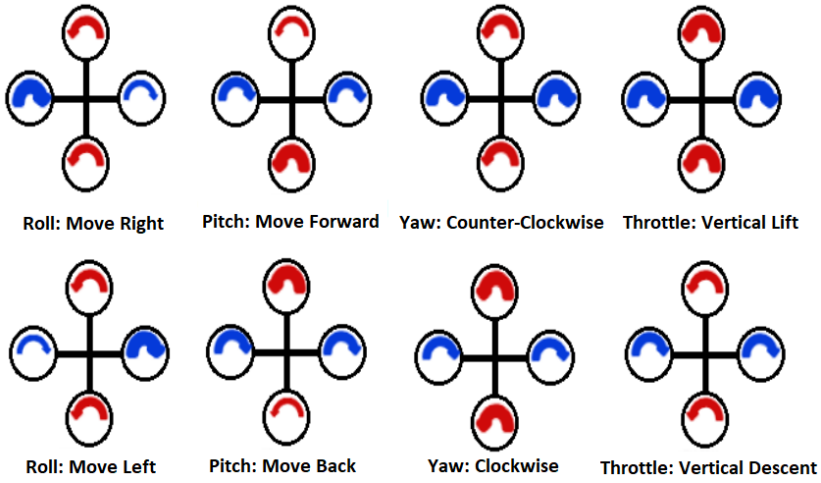


Figure 3: How control is achieved on a Quadrotor

A quadcopters structure is made up of a rigid, lightweight frame with four brushless motors attached to drive the propellers. The each of the rotors operate by a central microcontroller and powered by an onboard battery. Feedback control is provided by an IMU and communication is generally made using an RC Transmitter. Multiple open source software options are available to drive the quadcopter such as OpenPilot [23] or ArduPilot [8].

2.1.2 Ultrasonic Sensors

Ultrasonic sensors are widely used devices for mobile robotics. They consist of an emitter and a detector to send and receive signal waves respectively. Sensors calculate the time delay between sending a signal and receiving an echo, allowing for a distance measurement to be implied from this.

Although there are many benefits with ultrasonic sensors, most prominently the low cost, there are also disadvantages that must be considered [24]. A common problem with using ultrasonic sensors is specular reflection. Commonly in operating environments surfaces generally have smooth textures, which has the adverse effect of ultrasonic waves being reflected similarly to a mirrored surface. Simply put, having a greater incident angle to a surface incurs a smaller receiving signal. Another issue is the relatively slow propagation velocity of ultrasonic signals in air, approximately 343m/s. For instance, it would take an ultrasonic sensor approximately 50ms to measure a distance of 8.5m, which is a significant

period of time for real-time systems.

2.1.3 Laser Range Finder

A laser range finder uses a laser to measure the distance to a target object. A laser beam is sent from the sensor towards a target and the time of flight is measured. Multiplying this by the speed of light provides the distance to the target. Using a laser range finder provides a highly directive, monochromatic source while maintaining minimal power consumption. These features are important to improve signal to noise ratio, ensure the accuracy of the measurement and to elongate operation time. [25]

The core benefits of using a laser range finder are their accuracy and high efficiency in measuring objects even at a large incident angle. This allows for reliable results over a variety of potential surfaces which could vary in material type or incident angle relative to the sensor.

The major drawbacks of these sensors are attributed to their sensitivity to environmental conditions. Laser range finder measurements are commonly susceptible to corruption in environments with high mist, large amounts of dust or strong sunlight [26], although significant progress had been made to reduce these errors over time [27]. These conditions can disperse or reflect the signal prematurely, providing incorrect measurement results. Another downside of this type of sensor is that they cannot range transparent materials as the incident beam passes directly through them.

2.1.4 IMU

An IMU is a device that is mainly used to measure orientation, acceleration and gravitational force [28]. Early on IMU technology consisted of only an accelerometer and a gyroscope to measure inertial accelerations and angular rotation respectively. As IMU technology has advanced a third sensor type was introduced – a magnetometer which measures the bearing magnetic direction and so improves the gyroscopes readings.

IMU's are highly useful devices that are used consistently used within UAV development, but there are some limitations that must be considered. A key issue that is encountered when using IMU's for navigational purposes is the accumulation of error over time. Gyroscopes are known for their drift errors over a long intervals of time and integrating recorded accelerations from accelerometers can also quickly accumulate undesirable errors [29]. Accelerometers are particularly sensitive to accelerations when they have rapid rotational or translational motion [29].

A common approach to alleviate these issues is implementing the use of a Kalman Filter or an Extended Kalman Filter. Their implementation can be complex to begin with, but have proven results based on wide usage in many

related areas [28, 30, 31].

2.2 Review of Wall Following Techniques

Throughout prior literature various wall following techniques have been utilized in a number of different robotics projects. In particular the wall-following techniques being reviewed are those that pertain to unknown and unstructured environments where sensors must be used to determine the surroundings and analysis must be achieved in real-time to determine local trajectory of a system. The key areas that were explored and researched include map building, obstacle avoidance and improvement of position estimates.

Turennout, Honderd and Schelven investigated the use of basic ultrasonic distance measurements to determine the distance and orientation of a robot relative to a surface [32]. They use the sensor data to find estimates of the distance and orientation of the robot, which are then used within the feedback controller. A dead reckoning algorithm is also present, which is swapped to when the reference surface is no longer available. They discovered that within their experiments that the robot would have an error of only a few millimetres from the desired distance from the wall. This was deemed to be a highly effective technique for ground based projects which has been expanded upon with many future papers.

Nugraha Et Al looked into a state based approach in regards to wall following, attempting to stay close to a wall without turning into it [33]. It divides the wall into a series of action states that activate immediately based on incoming sensor values. Controlled testing of this state based system provided a successful result, with the robot managing to follow a left-most wall while not crashing into the wall or moving too far away.

Bemporad, Marco and Tesi developed a sensor fusion approach for the estimation of a robot's coordinates with the use of an Extended Kalman Filter [34]. Their development robot was equipped with 5 sonar sensors equally placed around a 180 degree angle, but only one was used for the initial testing. Experimental results showed that errors on the initial orientations can be up to 10-15 degrees are tolerable, while any larger errors would cause the sensor to lose contact with the wall.

Liu Et Al proposed a novel wall-following technique for mobile robots using a singular distance proximity sensor [35]. The benefits of this include the avoidance of interference between sensors, reduction of hardware costs and a robustness to external interference. They utilize a proximity control strategy to maintain a fixed distance between the wall and the proximity sensor. When the distance is too large the robot is rotated towards the wall and when the distance is too small it rotates away. This produces an oscillatory behavior which slowly self-converges to a steady fixed distance state. Results of both simulated and experimental tests show that their proposed algorithm worked effectively on both straight lines and simple arcs. This method can be seen visually in Figure 4.

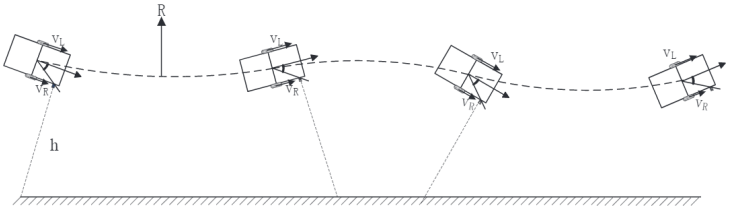


Figure 4: Singular proximity sensor wall following behaviour [35]

Imhof Oetiker and Jensen also explore a singular sensor approach [36]. They further this method by implementing a kalman filter for localization of the robot, yielding reliable wall following control method even on plane surfaces or in presence of large perturbation forces. They postulate that although fusing multiple sensors together provides a more accurate and robust system, using a single sensor still allows for the estimation of the necessary values.

An alternate approach is presented by Ando and Yuta [37], in which they use a wall-following sonar-ring to control an autonomous mobile robot. Within their experiments they used a 12 directional sensor array and an on-board controller which decides the motion of the robot, an example of which is shown in Figure 5. From their initial simulations they found that robot can track many varied surfaces as long as the wall is in range of detection, and that the wall following showed robust and reliable tracking. Fazli and Kleeman [38] also explored using a sonar ring, with a more complex setup of 48 ultrasonic transducers, 24 being receivers and 24 being transceivers. This provided accurate 360 coverage all around the robot when firing all transmitters. This was experimentally demonstrated and showed the effectiveness of obstacle avoidance and wall following algorithms with a simple and reliable implementation.

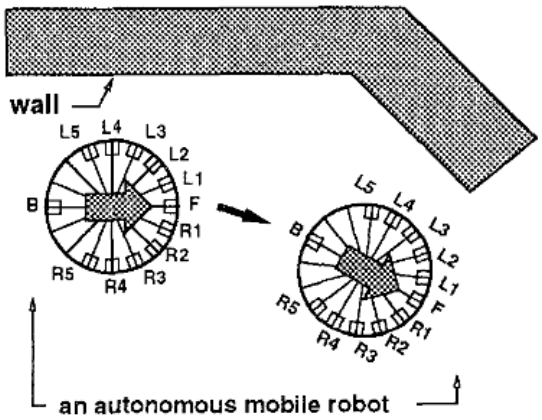


Figure 5: Autonomous robot detects and avoids front-left surface was using a sonar-ring [37]

Lynen Et Al investigate the use of a Multi-Sensor-Fusion Extended Kalman

filter, demonstrating the result on a MAV (Micro-Aerial-Vehicle) [39]. This framework allows for the accurate prediction of a system's position that is extendable to a large number of sensor inputs and robust to lost signals. They actualize this successfully on a MAV utilizing visual, inertial and pressure sensors alongside a GPS receiver.

Developing a UAV capable of performing inspection tasks within enclosed industrial environments is completed by Nikolic Et Al [40]. Within this it was proposed to control the system using an onboard IMU alongside a pair of cameras in stereo configuration. Using a model-predicative controller the UAV could fly in close proximity to surfaces while maintaining efficient trajectory following. This system worked very effectively, and comparing with a base truth it it was found that the drift rates varied only by a few centimeters in a typical flight period. This suggests that when using this method, following longer paths is feasible without relying on any external inputs.

2.3 Review of Simulation Methods

Many prior researchers have developed methods to simulate the operation of a UAV. This allows for testing of control algorithms before spending time and money developing the actual hardware and software used for the physical system. This is also valuable for rapid prototyping of various control algorithms which can be discarded if ineffective.

Comprehensive research has been completed into the area of modelling and identification to form the basis of a simulation. There are several different methods that can be used to formulate the equations of a ridged body with 6 DOF. A common approach for formulating the equations of motion is using Newton-Euler method [41–43]. These are generated in a body frame of reference to provide several benefits, including the inertia matrix being time-invariant, symmetry simplifying equation complexity and control forces being provided in the body frame [41]. These provide an effective and simplistic model of the system which compared well to an actual quadrotor [42], but do not account for any intricate dynamics of the system. Alternatively a similar method is utilized by Patel Et Al via a Lagrangian approach to form the simplified model [44].

Although simplistic models can be useful, they do not account for and model all of the dynamics that a UAV presents. Further research has been completed to attempt to encapsulate these effects within the dynamic model to allow for more accurate simulation. One such area to improve modelling is within the rotor dynamics. A simple model is shown by Sarim Et Al calculating the generated force as a function of the rotational speed of the motor and a thrust constant, determined experimentally [45]. A model with more complexity is presented by Mahony Et Al where the steady state thrust is calculated from the rotor as a function of angular velocity, density of air, thrust coefficient, radius and rotor disk area [46]. However, Mahony also recommended in practice that lumping all those terms together into a singular constant can be more effective. The benefits of determining this experimentally is that it can naturally incorporate the effect of drag induced by the rotor.

When rotors have translational motion horizontally through the air, the advancing blade has a faster tip velocity than the retreating bade which creates a difference in blade lift. This in turn causes the rotor to tilt in the direction of motion, known as Blade-flapping. This can be seen visually in Figure 6. Pounds Et Al model this disturbance by generating blade flapping equations as instantaneous functions of the systems velocity [47]. They determine the resulting angle by solving the constant and sinusoidal components of the blade centrifugal-aerodynamic weight moment system. These effects of Blade-flapping leads to an undesired introduction of induced drag.

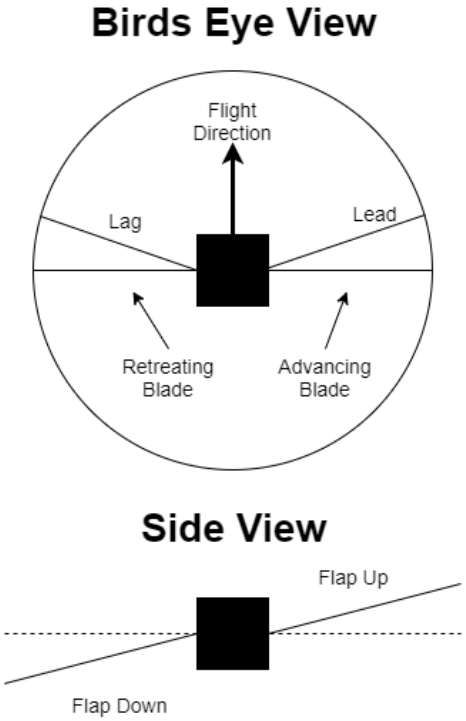


Figure 6: Visualization of the blade flapping phenomenon

With any generated lift there is an associated induced drag caused by the backward inclination of aerodynamic force with respect to the motion. While Blade-flapping is due to the flexing of rotors, induced drag is associated with the rigidity of the motor. Even so, Mahony Et Al [46] states that their mathematical expressions are equal and a single term is suffice enough to represent both effects as a lumped parameter in the dynamic model.

Depending on the desired application, there are also some optional additions that can be added to the overall dynamic model. One common addition is modelling natural atmospheric disturbances such as wind, which can be categorized into deterministic and stochastic models [48]. Examples of deterministic models include head wind, tail wind, cross wind and wind shear. By comparison,

the stochastic model covers fluctuations in wind translational and rotational velocities about certain pre-defined means. These are modelled as a stochastic process, such as von Karman or Dydren wind turbulence models [48]. Sharifi and Nobahari extended this modelling method by using a multiple model filter [49]. This considered 3 different wind models with an extended kalman filter to simultaneously estimate the wind. Results of this showed good performance and the proper estimation of the wind model.

There are multiple programming languages available to implement the full simulations such as C, MATLAB or Java. Realistically any language can be used so determining the most effective language is important. A commonly used simulation environment was using MATLAB Simulink [42]. This provides a graphical editor for modelling and simulating dynamic systems. This simulaton environment provided effective results when compared experimentally to real-life flight data. It was found that even with some minor oscillations the quadrotor closely matched the simulation as predicted [42].

Kaidi Et Al looked into designing a hard real-time UAV simulation system based off a real-time operating system [50]. They firstly developed a simulation model using MATLAB and Simulink, then transformed this in C code files directly. They divided up the simulation system to different tasks, including system management, model calculation, scheduling management and interface. It was compiled by Simulink Coder (Formerly Real-Time Workshop) and run on VxWorks RTOS. Simulation results showed that they could still meet the accuracy and real-time requirements. Overall once completed this method was found to accelerate the creation of a hardware-in-the-loop simulation system, reduce development difficulty and increase expandability.

Meyer Et Al designed a comprehensive simulation that integrates with ROS (Robotic Operating System) and Gazebo, a popular robotics simulation tool [51, 52]. This allowed for the simultaneous simulation of different aspects such as flight dynamics, on-board sensors, imaging sensors and complex environments. However, while their implementation covers gravity, contact forces and frictions, it does not cover aerodynamics and propulsion systems in its current form.

2.4 Review of UAV Control Utilizing Proximity Sensors

An investigation was completed into the state of the art research regarding UAV control utilizing proximity sensors. This provides a basis for this thesis showing what areas could be explored further.

One method suggested was to equip four proximity sensors on to the four sides of the UAV to provide the distances to the nearest obstacles in their respective directions, shown in Figure 7 [45]. This would allow the UAV to move around an enclosed area avoiding obstacles and align itself such that a wall is to it's left then maintain a fixed distance. Although theoretically this algorithm should work effectively, this method was only ever simulated and is untested on a physical UAV. This algorithm presented also has limitations in that it relies

on all the sensors providing accurate data and that there be surfaces available for ranging on all sides of the system.

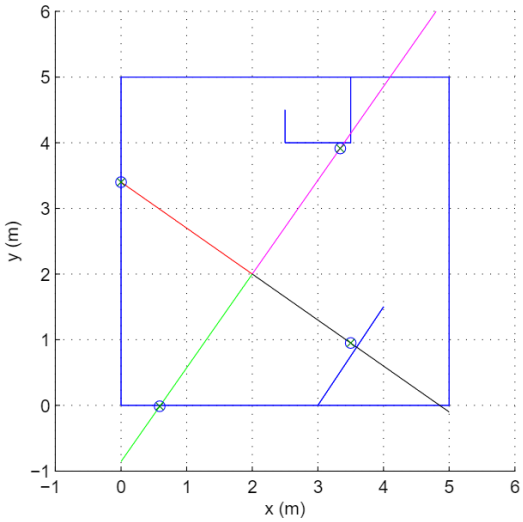


Figure 7: Visualisation of proximity sensors providing distances to enclosed room surfaces [45]

Another strategy explored is through the use of a scanning laser range finder which can measure the distance from the surrounding objects in a frontal 270 degrees and 30m range [53]. These laser range finders provide higher precision data compared to ultrasonic and infrared instruments. All measurements acquired from the laser scanner are treated as an obstacle and straight line fit is attempted to keep the UAV at a fixed distance from the detected surface. They conclude that this is an effective method that is applicable to multiple forms of UAV systems, with the main advantage being minimal required computation power.

Roberts Et Al explores a quadrotor system capable of autonomous operation within obstacle free indoor environments, achieved using more simplistic sensing and control strategies [54]. They use a mix of infrared and ultrasonic sensors mounted onto a custom quadrotor platform and they undergo experimental testing in an enclosed 6mx7mx3m space. They implement collision avoidance by calculating the difference in distance between opposing walls, then feeding this into the control to alter the orientation of the system to navigate away from a wall.

Grzonka Et Al developed a general navigation system which enables a small quadrotor system to autonomously operate in indoor environments [55], achieved by systematically extending and adapting techniques used by ground based robots. They describe a complete navigation solution that incorporates mapping, localization, path-planning and control. Their central method for navigation incorporates both SLAM algorithms and obstacle avoidance using an

on-board scanning laser rangefinder. The largest drawback discovered using their methodology was the high computational cost of their algorithms with extra computation having to be completed externally on a laptop computer.

One paper within the literature by Sa Et Al explored a method of shared semi-autonomous control for building inspections [11]. A method was presented using shared autonomy control to enable a low-skilled operator to be able to fly near a structure safely. This was achieved by reducing the number of degrees of freedom a operator needs to control down to 2 from 6. By creating a map of where they believe the UAV is currently situated in the space, they reduce the control to distance from the object and angle around the object (for this example a circular structure was used). This method was deemed effective, but had some minor drawbacks. Having a dedicated communications link is complex, limits range and can be unreliable; but is necessary as the calculations for this method have to computed remotely due to processing limitations on the UAV. This method was also tuned to be specific to a pole type structure, as is not applicable to a variety of applications and thus is not easily extendable.

2.5 Summary

This chapter has provided relevant background research and and an overview of current literature relating to the goals of this thesis. Background information on quadrotors and potential hardware is presented, providing functional overviews and brief discussion on advantages and drawbacks to consider. Relevant prior research is investigated, showing current techniques used for general wall following and simulation methods. A more focused exploration into control using proximity sensors is then completed, showing what current methods have already been developed. This information serves as a basis to identify what improvements that can be made to contribute further to current literature.

3 Simulation of quadrotor system

This chapter discusses the method formulated to simulate the basic dynamics of a quadrotor system, with additional on-board virtual sensors. The desired requirements are outlined and general software plan is introduced. Detailed equations are presented providing information on the implementation of the dynamic model, the surface model and the virtual sensors. Functional tests are completed to ensure the completed simulation software works accurately and without error.

3.1 Requirements

The simulation to be created requires certain properties and features to be successful. These requirements serve as the baseline for what needs to be implemented for the simulation to be deemed successful.

- Accurately simulate the dynamics of the quadrotor
- Accurately control the quadrotor through use of PID techniques.
- Model a varied surface for the quadrotor to track
- Model various sensors and their intersection with the aforementioned varied surface

3.2 Simulation Setup

3.2.1 Software

MATLAB was selected as the chosen software to implement the simulation within. MATLAB is a numerical computing environment that uses a proprietary programming language developed by MathWorks. It allows for simple implementation and development for analyzing data, creating algorithms, or creating models [56]. The high-level language supports numerical and symbolic calculations, and has vast array of mathematical functions and libraries available for use. These features that MATLAB provides makes it an ideal environment for implementing and developing the simulation alongside the ample tools available for subsequent testing after completion.

3.2.2 Fixed Constants

Various fixed constants are required to be determined for simulation of the quadrotor system. Constants relating to the quadrotor were measured experimentally or taken from relevant datasheets provided with the system. These particular values were based off the UAV selected for this project, the DYS 320 Full Carbon Fiber Folding Quadcopter, and could differ for alternate UAV systems. This provided the following fixed values for use on the UAV simulation which can be seen in Table 1

Table 1: Fixed UAV Constants

Name	Symbol	Value	Unit
Gravity	g	9.81	ms^{-2}
Mass	M	0.678	kg
Max Rotational Motor Speed	w_i	2890.3	rad/s
Thrust Constant	k_f	6.11e-8	N/RPM^2
Moment Constant	k_m	1.5e-9	Nm/RPM^2
Length from Rotor to Center of Mass	l	0.2	m
Moment of Inertia about X-axis	I_x	8.1e-3	kgm^2
Moment of Inertia about Y-axis	I_y	8.1e-3	kgm^2
Moment of Inertia about Z-axis	I_z	14.2e-3	kgm^2

3.2.3 Overview

The simulation was set up as a simple loop, which would iterate through the relevant dynamics and motor control of the UAV updating the position and angle in small steps. A certain time step was chosen indicating how many seconds to advance each loop. This was ideally kept at a low value to keep the simulation accurate, at the expense of longer computation time. A period of time for the simulation to run for was also indicated, giving the total number of iterations to loop through, as shown in Equation 1

$$iterations = simulation\_period * (\frac{1}{dt}) \tag{1}$$

Each iteration of the simulation involves the following general steps:

1. Transform sensor readings into a PD control output
2. Calculate the desired rotor speeds from control output
3. Calculate relevant dynamics of the system
4. Integrate accelerations to find velocity and displacements.
5. Simulate the sensor readings relative to the UAV

Note that the first loop through requires some initial dummy values until the dynamics and sensors have been simulated for the next iteration.

3.3 Dynamic Model

Within prior literature there can been multiple different methods used to dynamically model the a quadrotor system [57–59]. These range from basic, generalized models and to more advanced models that accurately reflect smaller disturbances and dynamics. For the development of this particular simulation a more generalized model of the quadcopter dynamics is desired as the focus is on sensor implementation as opposed to subtle disturbances or external forces. With these factors considered, the dynamic model chosen was based off prior work completed by Sarim Et Al and Bresciana Et Al [41, 45].

A free body diagram was created detailing the various forces and coordinate systems present. A visual representation of the diagram created can be seen in Figure 8. The Earth Frame (EF) shows the fixed reference frame in which all motion can be referred to globally. The Body Frame (BF) is the frame attached to the centre of mass of the UAV. A vertical thrust force is generated by each individual rotor on the UAV (F1-F4). In addition to this, each rotor also produces a moment perpendicular to the plane of propeller rotation (M1-M4). A downwards vertical force is also enacted onto the UAV representing weight, which is the product of the mass of the UAV (m) and earth's gravity (g).

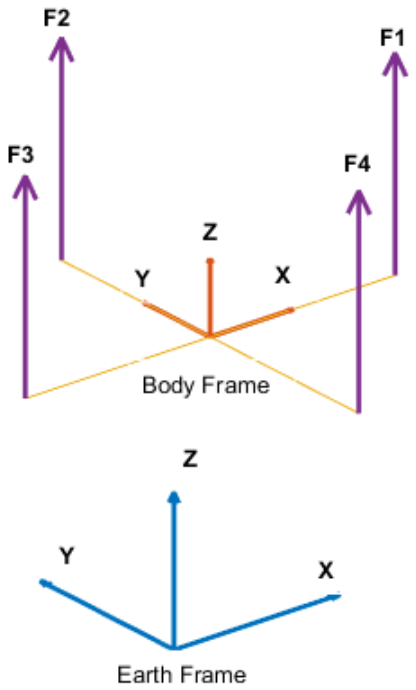


Figure 8: UAV Free body diagram

The moments produced by each individual rotor are directly opposite to the rotation of the propeller. These generate a rotation force around the Z axis, and so to cancel these out rotor's 1 and 3 and set to move in the clockwise direction while rotor 2 and 4 are set in the clockwise direction.

A rotation matrix is also desired to be formed to move between the Earth Frame and the Body Frame. To get the frame coordinates from the Body Frame to the Earth Frame, there is first an angular rotation around the X-axis (roll), followed a rotation around the Y-axis (pitch) and finally a rotation around the Z-axis. These angles are noted as phi (ϕ), theta (θ) and psi (ψ) respectively. Thus the rotation matrix formulated as shown in Equation 2.

$$\begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + s\phi s\psi c\theta \\ s\psi c\theta + s\phi c\psi s\theta & s\psi s\theta & s\psi s\theta - s\phi c\psi c\theta \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{2}$$

Where $c\phi$ and $s\phi$ signify $cos(\phi)$ and $sin(\phi)$ respectively, and similarly for other angles.

The position of the centre of mass in the earth frame is denoted by the vector r. This can be seen in Equation 3.

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \Sigma F_i \end{bmatrix} \tag{3}$$

Where m is the mass of the UAV, g is acceleration due to gravity, and F_i (i = 1, 2, 3, 4) is the force from a specific rotor I, given by Equation 4.

$$F_i = k_f w_i^2 \tag{4}$$

Where w_i is the angular velocity of the i^{th} rotor and K_m is a constant.

Using the knowledge of the rotor forces acting upon the system the angular acceleration of the UAV can be determined through use of Euler equations. These are detailed in Equation 5.

$$\begin{aligned} I_x\ddot{\phi} &= l(F_2 - F_4) \\ I_y\ddot{\theta} &= l(F_3 - F_1) \\ I_z\ddot{\psi} &= M_1 + M_2 + M_3 + M_4 \end{aligned} \tag{5}$$

Where l is the distance from the each rotor to the UAV's centre of mass. I_x, I_y and I_z are the moments of inertia along the x,y and z directions respectively. F_i (i = 1, 2, 3, 4) is the force from a specific rotor i, given by Equation 4. M_i (i = 1, 2, 3, 4) are the rotor's moments generated from the angular velocity of the rotors

$$M_i = K_m\omega_i^2 \tag{6}$$

Where ω_i is the angular velocity of the i^th rotor and K_m is a constant.

The acceleration of the centre of mass found in Equation 3 can be integrated to find velocity and displacement of the UAV. Furthermore using the angular accelerations calculated in Equation 5 the angular velocities and displacement can

be solved for again using integration. These can be completed in the simulation using the following formula in Equation 7.

$$\begin{aligned} \dot{\phi} &= \dot{\phi} + \ddot{\phi}dt \\ \phi &= \phi + \dot{\phi}dt \end{aligned} \tag{7}$$

Where dt is the time step of the simulation. ϕ can be interchanged similarly for x, y, z, θ and ψ

3.4 Surface Modelling

3.4.1 Implementation

The core of this thesis is focused around tracking varied surfaces and so this was required to be modelled within the simulation. This was desired to be highly adaptable so that many different surface profiles could be simulated accurately. This would allow for many different possible situations to be tested with various tracking algorithms before implementation on a real-world UAV.

This surface was chosen to be modelled as series of coordinates in 3-space (X,Y,Z) forming a surface plane. Each coordinate is stored in a 2D array, referenced by their i,j locations. All Y and Z coordinates were chosen to be placed at a fixed spacing, with the X value being interchangeable to allow for desired variations to the surface. This method of storing this plane can be visually seen in Figure 9.

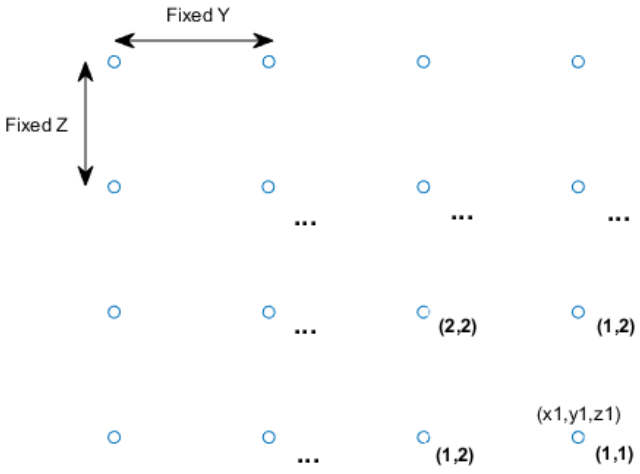


Figure 9: Modelling surface plane structure

Where (1,1) , (1,2) etc are the i,j, reference location and each blue dot representing an coordinate in 3-space (i.e x1,y1,z1).

Using a MATLAB script these surface coordinate can be automatically generated to form the required plane. An example of one such script can be seen in Figure 10, where the X coordinates have been randomly generated to have a slight offset to create a varied surface. This is only one such possibility and differing scripts can been written to create any surface profile desired.

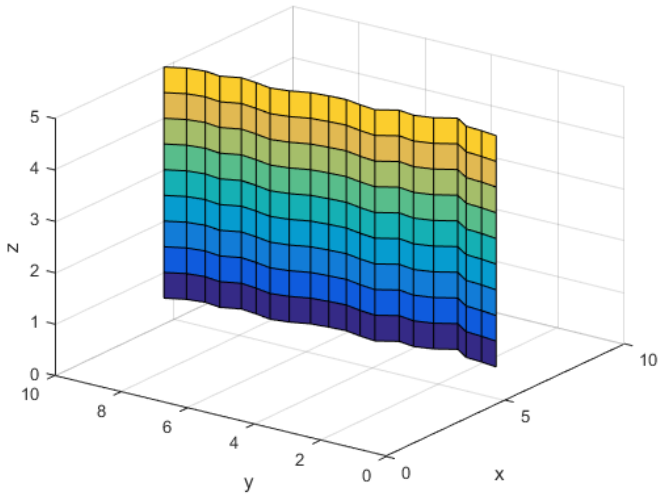


Figure 10: Example of a modelled surface plane

3.5 Sensor Modeling

A requirement of the simulator is to virtually track a surface using proximity sensors. The first step towards this is to model these sensors and mount them virtually onto the simulated UAV.

3.5.1 Initial Placement

The sensors need to be first placed at fixed location on the dynamic UAV model. This placement is not always a fixed XYZ vector relative to the centre of mass, as angular rotations also can modify the placement and rotation of the sensors.

A vector is firstly used to indicate the location of the senor relative to the centre of mass of the UAV (v_{sl} = vector sensor location). This vector is in the body frame of the UAV, and specifies the X,Y and Z placement distances of the sensor. Secondly a second vector is used to indicate the direction that the sensor is facing, with the length of the vector also indicating the maximum sensor range. These two vectors are shown in Equation 8.

$$\begin{aligned} v_{sl}^{BF} &= \begin{bmatrix} x & y & z \end{bmatrix} \\ v_{sd}^{BF} &= \begin{bmatrix} x & y & z \end{bmatrix} \end{aligned} \tag{8}$$

Both of these vectors are required to be transformed into the earth frame based on the current location and angle of the UAV. To help achieve this goal firstly the euler angles of the UAV were converted into a quaternion shown in Equation 9.

$$q = \begin{bmatrix} cos(\frac{\phi}{2})cos(\frac{\theta}{2})cos(\frac{\psi}{2}) + sin(\frac{\phi}{2})sin(\frac{\theta}{2})sin(\frac{\psi}{2}) \\ sin(\frac{\phi}{2})cos(\frac{\theta}{2})cos(\frac{\psi}{2}) - cos(\frac{\phi}{2})sin(\frac{\theta}{2})sin(\frac{\psi}{2}) \\ cos(\frac{\phi}{2})sin(\frac{\theta}{2})cos(\frac{\psi}{2}) + sin(\frac{\phi}{2})cos(\frac{\theta}{2})sin(\frac{\psi}{2}) \\ cos(\frac{\phi}{2})cos(\frac{\theta}{2})sin(\frac{\psi}{2}) - sin(\frac{\phi}{2})sin(\frac{\theta}{2})cos(\frac{\psi}{2}) \end{bmatrix} = \begin{bmatrix} q1 \\ q2 \\ q3 \\ q4 \end{bmatrix} \tag{9}$$

Where ϕ = roll, θ = pitch and ψ = yaw

Using this quaternion the rotations of any vectors can be simply calculated given the Euler angles of the UAV. Both the sensor direction and position vectors are rotated by this quaternion to provide new vectors indicating the sensor direction and position in the earth frame. This is achieved using the following method.

Firstly the quaternion is converted to a direction cosine matrix.

$$dcm = \begin{bmatrix} q_3^2 + q_0^2 - q_1^2 - q_2^2 & 2(q_0q_1 + q_3q_2) & 2(q_0q_2 - q_3q_1) \\ 2(q_0q_1 - q_3q_2) & q_3^2 - q_0^2 + q_1^2 - q_2^2 & 2(q_1q_2 + q_3q_0) \\ 2(q_0q_2 + q_3q_1) & 2(q_1q_2 - q_3q_0) & q_3^2 - q_0^2 - q_1^2 + q_2^2 \end{bmatrix} \tag{10}$$

The direction cosine matrix is then multiplied by sensor direction and location vectors to return a newly rotated vector.

$$\begin{aligned} v_{sl}^{EF} &= dcm * v_{sl}^{BF\prime} \\ v_{sd}^{EF} &= dcm * v_{sd}^{BF\prime} \end{aligned} \tag{11}$$

Finally the sensor location vector is added to the current position of the UAV, providing the current positon of the sensor in the earth frame.

An example of this rotation method with two sensors mounted at a $\pm$0.1m offset along the y-axis of the UAV can be seen in Figure 11.

3.5.2 Intersection Algorithm

With the sensors orientated correctly within the simulation, the sensor functionality needs to be modelled. This requires each sensor to return a value indicating the distance that it is measuring. In free space this will be the maximum sensor measurement distance. If the sensor detection vector intersects with a surface before the maximum sensor distance, then the intersection distance will be returned instead. For this to be achieved an algorithm was created to determine the intersection of a vector with the surface model outlined in Section X.X

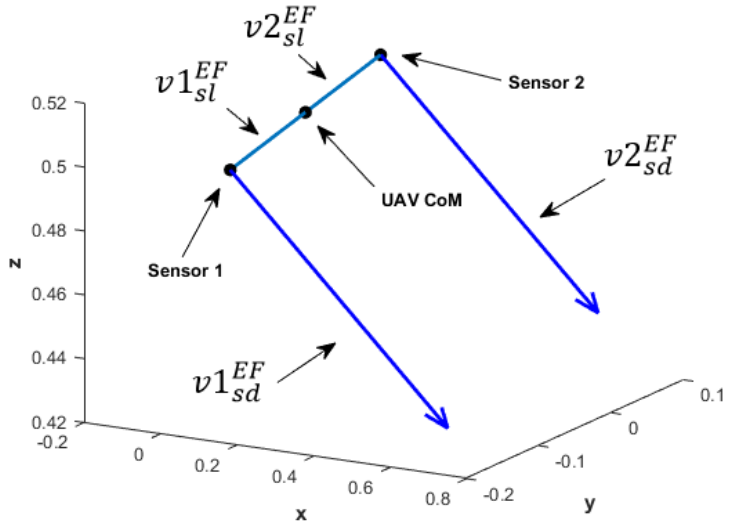


Figure 11: Rotation of 2 offset sensors along y-axis in body frame. Rotated by phi (ϕ) = 5°, theta (θ) = 5°, psi (ψ) = 5°

Two functions were implemented for this purpose. The first does a full iteration through the surface model array to find any intersection points, then selects the first intersection with the vector. The second uses prior knowledge, and only searches for intersections near the previous vector in the last time step. The former algorithm is far more robust, but takes a significantly longer computation time. The latter algorithm can fail in some circumstances, but is much faster. A mix of both algorithms are used in the final simulation to achieve optimal accuracy and speed. The following method is used to find an intersection between a vector and a plane bounded by 3 points.

Firstly, this intersection algorithms takes 3 coordinates to form a triangle in 3-space. These points are used to describe the plane in which the intersection with the sensor vector will be confirmed. Thus, these points are first converted to a plane described by a normal vector and a single point on the plane.

$$\begin{aligned} A &= \begin{bmatrix} x1 & y1 & z1 \end{bmatrix} \\ B &= \begin{bmatrix} x2 & y2 & z2 \end{bmatrix} \\ C &= \begin{bmatrix} x3 & y3 & z3 \end{bmatrix} \end{aligned} \tag{12}$$

$$u = B - A \tag{13}$$

$$v = C - A \tag{14}$$

$$w = u \times v \tag{15}$$

$$P = v1 \tag{16}$$

Where A,B and C are the 3 coordinates describing a triangular plane, w is the vector direction perpendicular to the plane and P is a point on the plane.

The sensor vector line can be described by two end points, P_0 and P_1 or alternatively using the line vector q. Using this alongside the newly described plane vector the intersection can be found between the two, assuming one exists. The sensor line vector can be described by the following parametric equation

$$P(s) = P_0 + s(P_1 - P_0) = P_0 + s * q \tag{17}$$

$$q = P_1 - P_0 \tag{18}$$

For any intersection calculation between a line and a plane, there either exists a single point or they are parallel. It is first checked whether the line and plane are parallel by calculating the dot product between w and q.

$$t = w \cdot q \tag{19}$$

If the result equals zero, then it indicates that the line direction vector u is perpendicular to the plane normal w. If this is true then the plane and the sensor vector either lie directly within each other or do not intersect at all. Determining whether the intersection is continuous or disjoint can be solved by finding whether any one specific point of the sensor vector is contained within the plane. r is calculated which describes the vector between a point on the plane and the start point of the sensor vector line. If r is zero, then it must lie within the plane.

$$r = P_0 - P_plane \tag{20}$$

$$N = -(w \cdot r) \tag{21}$$

If the line and plane are found not to be parallel then they must intersect at a singular point. At this unique point of intersection, $w + sI * u$ is perpendicular to w. Knowing this the dot product equation is formed as follows.

$$w \cdot (r + sI * u) = 0 \tag{22}$$

Solving this for sI the solution is found to be

$$sI = \frac{-(w \cdot r)}{w \cdot u} = \frac{N}{T} \tag{23}$$

Since the sensor vector is a finite length from P_0 to P_1, to verify that there is an intersection within this length a simple check is made to confirm that $0 \leq \text{sI} \leq 1$.

Finally using this knowledge the intersection point is found by the following equation

$$I = P_0 + sI \bigodot u \tag{24}$$

Finally the point of intersection is confirmed to be within the bounding area of the 3-points forming a triangle. This is the achieved through the use of Barycentric coordinates. Consider Figure 12 where A, B and C are coordinates defining a triangular surface and P is a point of intersection. Barycentric coordinates can be used to show the position of the point located with 3 scalars, α, β and γ.

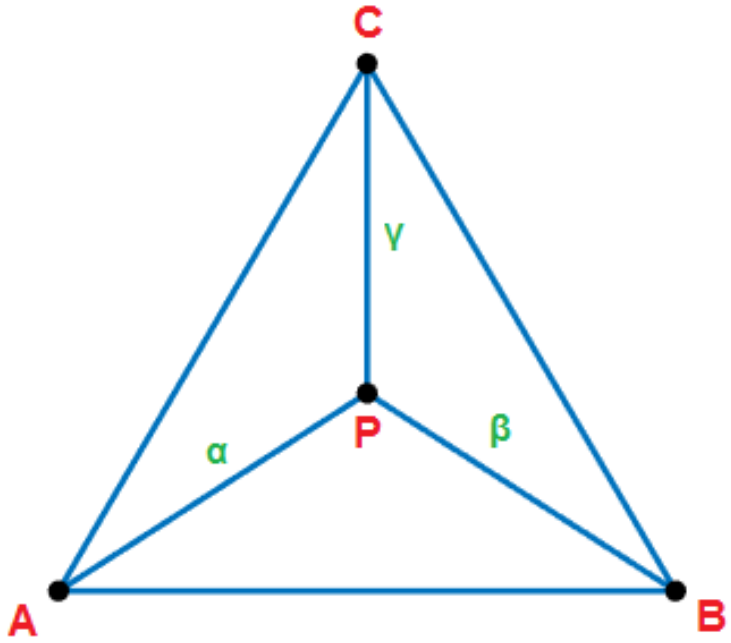


Figure 12: Barycentric Coordinates

The barycentric equation used to compute the position of intersection is as follows

$$P = \alpha A + \beta B + \gamma C \tag{25}$$

Where α, β and γ are three normalized scalars that sum to 1.

$$\alpha + \beta + \gamma = 1 \tag{26}$$

If any of the these Barycentric coordinates is not within the range of 0 to 1, then the point lies outside the triangle. This can be used to determine whether the point of intersection lies inside or outside the triangle.
Barycentric coordinates are defined in terms of area ratios.

$$\begin{aligned} \alpha &= \frac{PBC}{ABC} \\ \beta &= \frac{APC}{ABC} \\ \gamma &= \frac{ABP}{ABC} \end{aligned} \tag{27}$$

It can also be noted that the areas are proportional to cross products of the corresponding areas. An example of such a cross product calculations can be seen in Equation 28.

$$Area(PBC) = ||(C - B) \times (P - B)||/2 \tag{28}$$

Substituting in equations X,Y and Z provides the equation for calculating the barycentric coordinates of point P.

$$\begin{aligned} \alpha &= \frac{((C - B) \times (P - B)) \cdot n}{((B - A) \times (C - A)) \cdot n} \\ \beta &= \frac{((A - C) \times (Q - C)) \cdot n}{((B - A) \times (C - A)) \cdot n} \\ \gamma &= \frac{((B - A) \times (Q - A)) \cdot n}{((B - A) \times (C - A)) \cdot n} \end{aligned} \tag{29}$$

These equations can be further simplified before software implementation. Firstly with consideration of Equation 26, only two of the three unknown variables are required to be calculated using cross products. Secondly, the required vector calculations can first be simplified into a singular variables.

$$\begin{aligned} u &= B - A \\ v &= C - A \\ w &= P - A \end{aligned} \tag{30}$$

The denominator can also be simplified by calculating the normal of the triangle

$$\begin{aligned} n &= ((B - A) \times (C - A)) \\ n &= u \times v \end{aligned} \tag{31}$$

This substituting Equations 30- 31 into Equations 29 results in fully simplified equations to implement into the simulation.

$$\begin{aligned} \gamma &= \frac{n \cdot (u \times w)}{n \cdot n} \\ \beta &= \frac{n \cdot (w \times v)}{n \cdot n} \\ \alpha &= 1 - \gamma - \beta \end{aligned} \tag{32}$$

As outlined previously, if α, β and γ all lie within the range of 0 to 1 then the intersection point is inside the triangle, thus

$$\begin{aligned} \alpha \geq 0 \ \& \ \alpha \leq 1 \\ \beta \geq 0 \ \& \ \beta \leq 1 \\ \gamma \geq 0 \ \& \ \gamma \leq 1 \end{aligned} \tag{33}$$

These conditions are tested within the simulation to confirm whether the intersection point lies inside or outside the triangular bounding area.

3.5.3 Full Intersection Search

To initially find the point of intersection of a vector to the surface model, a brute force approach is taken. This algorithms takes the sensor location vector (EF), the sensor direction vector (EF) and the X,Y,Z coordinates describing the varied surface. The algorithm iterates through all possible i,j coordinates within the surface model, checking if there is an intersection in their respective region. A region is described as four adjacent coordinate points, forming a square. The index of each region is determined by the lower right hand coordinate. For each region there exists 4 possible triangles that could be intersected with. An example region can be seen in Figure 13, showing the different intersections possible. The intersection points for each triangular plane are determined using the intersection algorithm provided in Section 3.5.2.

From the four possible points of intersection, a single point must be selected as most accurate point of intersection for that region. This is because as local regions are the finest element possible in the surface model, there can only be one most accurate intersection point on them.

Firstly, any intersections that do not lie within the triangle plane region are discarded using the Barycentric method described in Section 3.5.2. If they do not intersect in the correct triangular region, they are too far offset from the plane and would give inaccurate results. In Figure 13 an example of such intersection points are indicated in (C) and (D).

Following this, the correct intersection location from the remaining valid points must be determined. The method to determine this was chosen as considering the angle of the triangular planes formed. The plane with the steepest angle is chosen to be the correct intersection plane.

The reasoning for this can also be seen visually in Figure 13 when comparing between (A) and (B). Both of these intersected with the vector but the region as a whole is sloping backwards towards the top right coordinate, indicating visually that (B) is more accurate. Whenever there is an outlying coordinate, it will produce a steeper plane and be more likely to be closest to the correct intersection location.

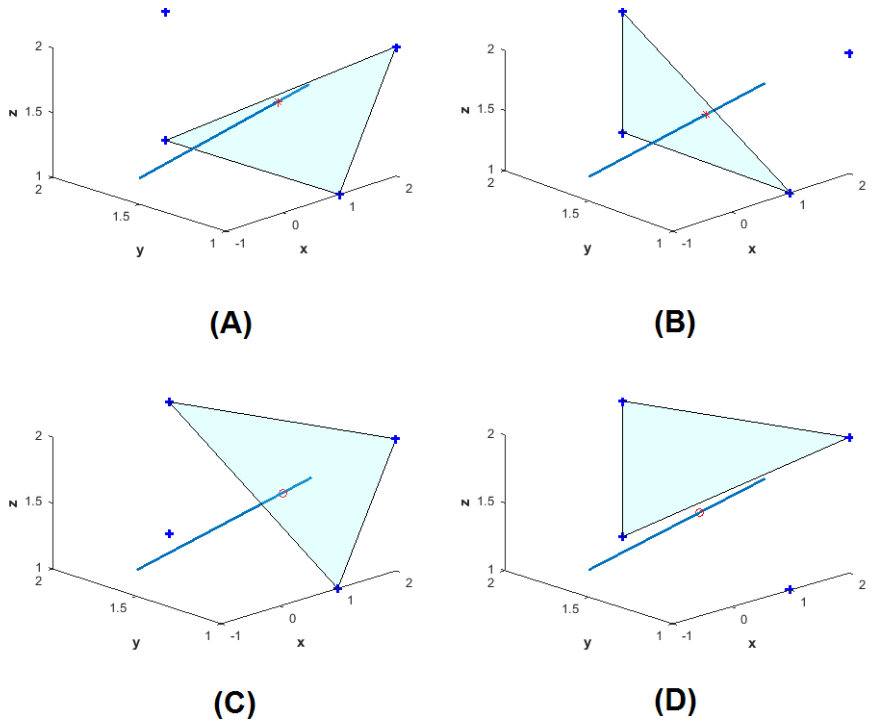


Figure 13: Plane intersections from four possible 3-point coordinate combinations. Red star where an intersection within the triangle occurs, red circle when the intersection with the plane is outside the triangle bounds.

The limits of this method begin to occur when all generated triangular planes are at steep angles. This indicates and irregular surface bounded by the 4 points that can give incorrect results. Using an appropriate small spacing for the surface modelling should minimize the effect of small intersection inconsistencies as it would still be close enough the optimal outcome, and can be treated as noise.

This method of determining intersections in each region is repeated across the whole surface model, until all points of intersection are found. The final problem to solve is finding which of the point of intersection the correct. In most cases, there would only be a singular point of intersection with the surface model, but if you considered a varied surface it is possible for there to be multiple intersections. An example of this is shown in Figure 14

In the case of multiple intersections, these are compared to find the closest distance to the initial sensor location. Multiple intersections can only occur when the sensor vector penetrates through the wall causing further intersections with the surface model. This implies that the closest point of intersection will always be correct.

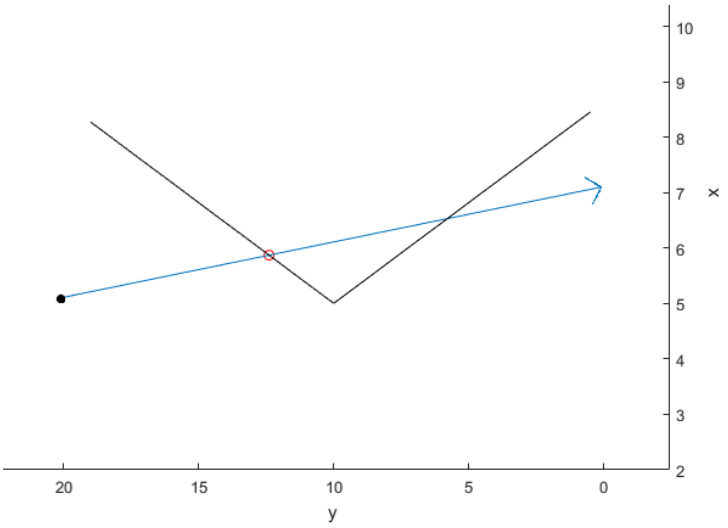


Figure 14: Double Intersection with surface model

The distance between the point of intersection and the sensor location is then returned providing a sensor reading. For purposes of optimization, the i,j coordinates of the intersection region are also saved for the next iteration for reasons discussed in subsequent Section 3.5.4.

3.5.4 Local Region Intersection

Using the full intersection search method as described previously would be highly inefficient as every local region would require to be iterated across on each loop. A secondary algorithm is proposed utilizing the prior intersection location to determine the next. This can be achieved assuming that a small enough timestep is used within the simulation such that the sensor vector would never jump a large distance over a linear surface model. Knowing this, it can be said that the next sensor intersection point is likely to be close to the prior intersection point. Using i,j coordinates indicating the previous region intersected with on the surface model, a local search can be completed nearby these coordinates as opposed to across all regions in the model.

This algorithm acts similarly to the full region intersection algorithm, but instead only checks the regions indicated by the adjacent i,j coordinates as opposed to iterating over all regions. This can be seen visually represented in Figure 15

3.6 UAV Controller

3.6.1 UAV Stabilization and Control

The UAV is controlled by 4 individual rotor working in conjunction to provide lift for the system. Thus 4 independent speeds for each rotor are required to be calculated to successfully operate the UAV. PD controllers were used as the

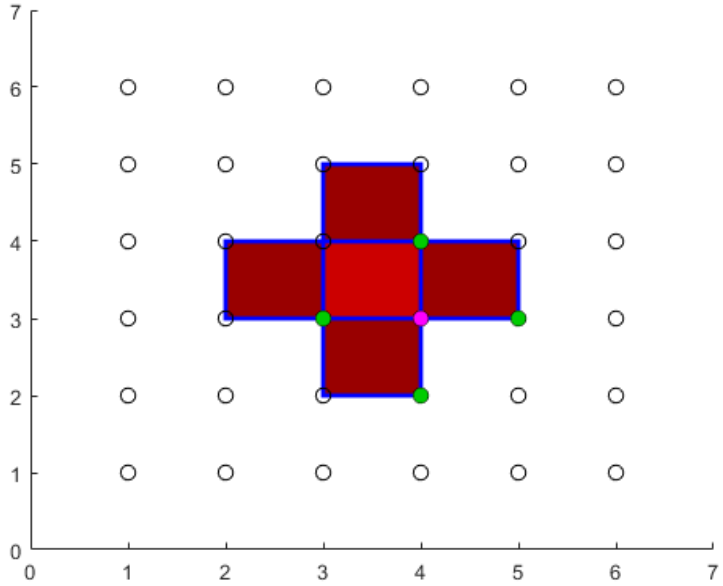


Figure 15: Visual representation of local region search. The pink point shows the previous i,j intersection coordinate with the red square indicating it's corresponding region. The green point designates the local adjacent i,j coordinates and their corresponding maroon regions.

method to determine the control outputs for each motor.

For initial implementation in the simulation the system was considered fully localized using direct data only available in the simulation. Note that this exact positioning of the UAV would not be as available in the real-world system and localization techniques would have to be further employed.

Individual PID controllers are used to generate command accelerations based off the deviations from the desired position and rotational state. The first 3 controllers designed are used to calculate x_{dd}, y_{dd}, and z_{dd}; the command accelerations. These are done using a simple PD controller, based off the approach taken by Tan and Kumar [60].

$$\begin{aligned} x_{dd} &= k_p^x * e_x + k_d^x * (\frac{e_x - e_x^{prev}}{dt}) \\ y_{dd} &= k_p^y * e_y + k_d^y * (\frac{e_y - e_y^{prev}}{dt}) \\ z_{dd} &= k_p^z * e_z + k_d^z * (\frac{e_z - e_z^{prev}}{dt}) \end{aligned} \tag{34}$$

Where e_i (i=xyz) is the error between the desired position and the current position of the system, e_i^prev is the error e_i found in the prior PD loop iteration,

k_p^i is the proportional gain, k_d^i is the derivative gain and dt is the simulation timestep.

Pitching and rolling the UAV is utilized to move along the x-y axes to the desired x-y position. To achieve this the command accelerations x_{dd} and y_{dd} converted to the desired angles θ and ϕ (pitch and roll).

$$\begin{aligned} \phi &= (\frac{1}{g})(x_{dd} * sin(\psi_{des}) - y_{dd} * cos(\psi_{des})) \\ \theta &= (\frac{1}{g})(x_{dd} * cos(\psi_{des}) - y_{dd} * sin(\psi_{des})) \end{aligned} \tag{35}$$

Where ψ_{des} is the desired yaw and g is the gravity.

To prevent the UAV from moving too quickly horizontally or exceeding stable angles a limit is implemented which simply prevents the calculated desired angles ϕ_{des} and θ_{des} from exceeding a set value.

The desired angles for pitch and roll are then used within a further PD controller to determine the control signals for each angle.

$$\Delta\omega_f = \frac{m}{8 * k_f * w_h} * z_{dd} \tag{36}$$

Where m is the mass of the UAV, k_f is the thrust constant, w_h is the nominal speed to hover in steady state and z_dd is the command acceleration in the z direction.

Using the control signals generated from each PD loop, each individual rotor speed is calculated. A vector can be written describing the desired rotor speeds as a linear combination of 4 terms.

$$\begin{bmatrix} \omega_1^{des} \\ \omega_2^{des} \\ \omega_3^{des} \\ \omega_4^{des} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \omega_h + \Delta\omega_f \\ \Delta\omega_\psi \\ \Delta\omega_\theta \\ \Delta\omega_\psi \end{bmatrix} \tag{37}$$

Where ω_h is the rotor speed required to hover in a steady state and $\Delta\Omega_f$, $\Delta\omega_\psi$, $\Delta\omega_\theta$, $\Delta\omega_\phi$ are the generated control signals from the respective PID control loops.

Following this calculation the rotor speeds must be limited to the maximum possible output of the modelled motors. This provides a more realistic representation of the system and prevents the simulated UAV from becoming unstable due to excessively high speed fluctuations in some circumstances.

Based off these desired rotor speeds, the lift force that the rotor produces is updated. Vertical force is produced according to following Equation 38.

Table 2: Variable Simulation Constants

Name	**Symbol**	**Value**
Timestep	dt	0.01
Simulation Period	t_{total}	18s
Hover Height	$z_{desired}$	2.5m
x Proportional Gain	k_p^x	3.1
y Proportional Gain	k_p^y	3.1
z Proportional Gain	k_p^z	3.1
x Derivative Gain	k_d^x	3.9
y Derivative Gain	k_d^y	3.9
z Derivative Gain	k_d^z	3.9

$$F_i = k_f \omega_i^2 \tag{38}$$

Where k_f is the thrust constant.

Each motor also produces a moment similarly

$$M_i = k_m \omega_i^2 \tag{39}$$

Where k_m is the moment constant.

The calculated forces and moments from this can then be used within the dynamics equations to simulate the movement of the UAV system.

3.7 Functional Testing of Simulation

Testing was completed to confirm the functionality of the simulator and determine the performance. All simulation testing was completed on a computer running Microsoft Windows 10 with an Intel Core i7-4790 CPU @ 3.60 Ghz.

3.7.1 Dynamic Model and Control Validation

Validation of the dynamic model and the system controller was completed to ensure correct functionality. The fixed UAV constants were set as per Table 1 in Section 3.2.2. Variable simulation constants set to the values shown in Table 2.

A fully localized system was used for experimental testing, made available within the simulation environment. This provided the current position of the UAV at any given time. Four waypoints were provided for the system to move between; and initial launching waypoint followed by 3 further waypoints to move between while airborne. The chosen values for this particular test can be seen in Table 3. Using the PD controller the system will attempt to track towards each of the waypoints in succession, and move to the next when it has come to rest within a tolerance of 0.01m.

Table 3: Navigation Waypoints

Launch	(X_0, y_0)	$(1, 1)$
Point 1	(X_1, y_1)	$(2, 1)$
Point 2	(X_2, y_2)	$(2, 4)$
Point 3	(X_3, y_3)	$(4, 4)$

The results of this can be seen in Figure 16. From this it can be clearly seen that the controller works correctly and tracks towards each waypoint in succession. From visual inspection, the dynamic model appears to be functioning correctly and compares similarly to results from Sarim Et Al [45]. These results are sufficient enough to meet the initial requirements discussed in Section 3.1.

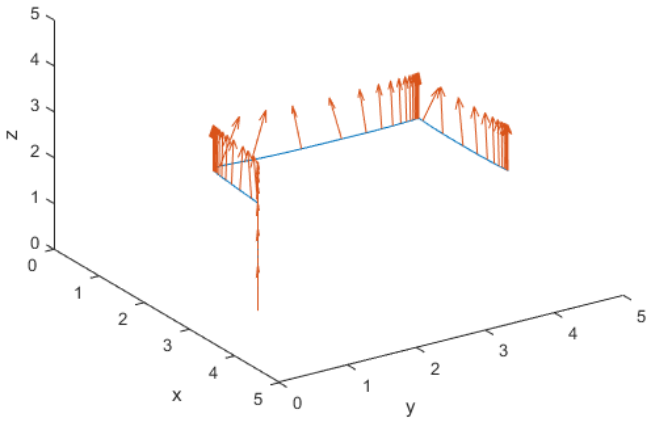


Figure 16: Simulation of quadrotor traveling between waypoints. The orange arrow indicates the local z direction of the quadcopter

3.7.2 Model Surface and Sensor Validation

Validation of the surface and sensor models are required to ensure they work without error.

A varied virtual surface was generated for the quadcopter to track using virtual sensors. A singular virtual sensor was placed onto the simulated quadcopter at the centre of mass, facing towards the surface. The same variable simulation constants are used from Section 3.7.1, as seen in Table 1. A varying model surface is generated with the parameters shown in Table 4, which randomly generates a small change in y-offset across the surface. The quadcopter controller is adjusted to first launch to the desired height following by tracking a fixed speed of 0.2m/s parallel to the surface. The results of this simulation can

Table 4: Parameters used to generate random varied surface

Width	5m
Height	5m
Spacing	0.2m
No. Rows	25
No. Columns	25
X Offset	4
Random X Variation	(0.1 < x < -0.1)

be seen visually in Figure 17

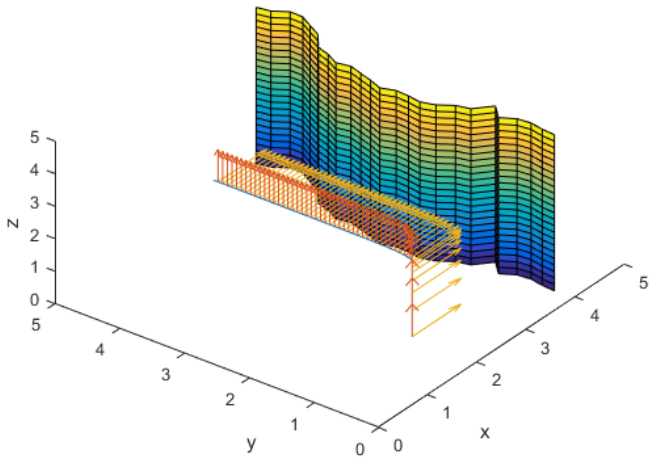


Figure 17: Simulation of a localized quadrotor moving horizontally at a fixed speed, while accumulating sensor data of profiled surface

The sensor data was then pulled from the simulation and compared to the varied surface. Immediate results showed that the sensor data precisely matched that of the actual distance to the surface. This is shown visually in Figure 18 where the recorded sensor data is offset to by 2m and can be seen to match the profile perfectly. This indicates that the implemented sensors work as expected within the simulator.

3.7.3 Performance

MATLAB provides the functionality to return a performance profile of the system, and determine the precise execution times within the simulator. From this it was ascertained what functions dominate the running time.

The simulator was set up with a flat plane of 5m x 5m, with a x-offset of 5m. The quadrotor system was set to launch to a height of 2.5m at coordinates $(3, 1)$

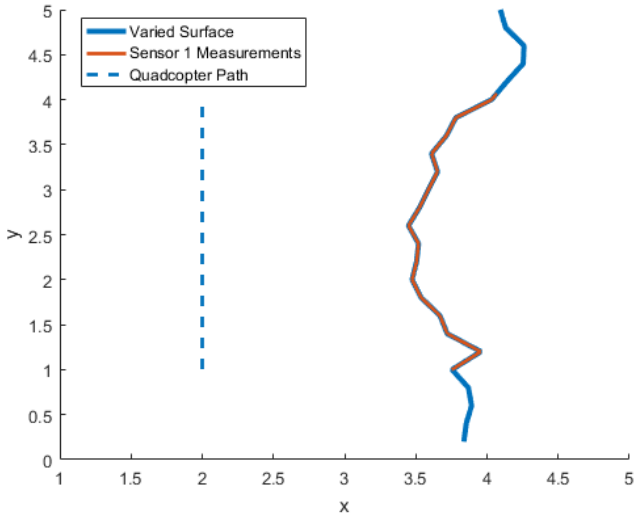


Figure 18: Comparison of measured sensor data to a profiled surface

Table 5: Summarized Simulation Performance Results (Full results provided in Appendix A)

Functions	**Total Runtime**	**Total Calls**	**Average Runtime**
Full Simulation	11.309s	N/A	N/A
Full Intersect Search	10.086s	632	16.0ms
Local Intersect Search	0.812s	1000	0.8ms
Other	0.411s	N/A	N/A

and then move towards navigation coordinate $(3, 4)$. The simulation timestep was set to 0.01s and run for 10 seconds total. The performance results of this simulation can be seen in Table 5.

Initial inspection of the performance profile immediately shows that the running time is dominated by the intersection search algorithms, currently taking up 96.4% of the total running time. This entails that in order for significant optimizations to be made, these algorithms should be targeted and modified first.

Further reviewing the two different algorithms used for intersection searches, it can be seen that the local intersection algorithm is 20 times faster than a full search. This value would scale even further as the number of surface model regions is increased. The local intersection search algorithms consistently check only 5 regions whereas the full intersection searches all regions.

It can be also seen that the full search algorithm is called 632 times, which only occurs when the local search algorithm fails to find an intersection. This gives the local search algorithm only a 36.8% success rate. From this it is immediately clear that improving the local search algorithm would reduce the overall

simulation time by minimizing calls to the full search algorithm. This could be achieved in future work by extending the local search area or implementing a new method entirely.

3.8 Summary

This chapter described the underlying methodology used to form the simulation software. Functional capabilities of the produced software include quadrotor dynamics, localized control and the addition of a virtual sensor surface detection. Testing of the completed system confirmed the functionality of simulation dynamics and the virtual sensors. Although the current rendition of the simulation operates accurately, it was identified that the performance of the system could be improved drastically.

4 Development towards UAV control methods using proximity sensors

The goal of this thesis is to implement autonomous algorithms for a quadrotor system using basic proximity sensors. Within this section, different algorithms are presented that can be used to help achieve this goal.

It is desired to test the different possibilities of using basic sensors in isolated cases, to assess their potential viability as an effective means of control. Due to this, development and testing are completed by developing automated control methods for a particular principle aircraft axis at any given time. The rest of the principle axes are controlled using localization data available within the simulation, allowing for more precise control that would be impossible in a real-world system. This allows for the development and testing of a singular principle axis while retaining the functionality of the entire system.

This chapter describes the various algorithms tested within the simulator for isolated control of a single channel. An overview of the simulated hardware is first provided, alongside the standard hovering algorithms to be used throughout the chapter. Considerations that must be made in regards to sensor placement on a quadrotor are presented. Algorithms are explored towards tracking flat surfaces via control of isolated channels, and these are validated within the simulation. This is extended further with development towards potential algorithms for profiled surfaces.

4.1 Initial Simulated Hardware

Making use of the developed simulation software in MATLAB, the first control algorithms can be produced. This will follow a basic 3-sensor setup; a singular downwards facing sensor to measure the height of the UAV and two horizontal sensors to detect the vertical surface. The new algorithms formed using virtual sensors replace the aforementioned control algorithms in Section 3.6.1, as they rely on a fully localized system. For simulation purposes, the algorithms produced do not need to be tuned to be as optimized as possible – but instead just tuned sufficiently to show that the suggested implementation would be possible.

4.2 Hovering Algorithm

As the first step towards autonomous control, a basic hovering algorithm was implemented to allow the UAV to rise to a desired height. The UAV uses the downwards facing sensor to monitor this height measurement. This was modelled in the simulation as a virtual sensor, which detects the distance from the UAV to the flat x-plane in the global frame.

A correctly tuned controller for altitude control can be difficult to achieve. small errors in sensor readings can easily make the quadrotor unable to maintain the desired height [61]. This should be considered for a real-world design, but poses

less of an issue within an error free simulation. Typically a PID controller or similar is used as the standard controller for altitude control [47, 62, 63].

A PD controller was selected as the method to allow the system to hover steadily at some desired height.

$$\begin{aligned} Z_{E\_PREV} - Z_E \\ Z_E = Z_{desired} - heightSensor \\ Z_D = \frac{Z_E - Z_{E\_PREV}}{dt} \end{aligned} \tag{1}$$

Where Z_E is the error between the desired height and the height sensor value, Z_{E_PREV} is the error of the previous iteration, Z_D is the derivate error based on the change in error and dt is the simulation time step.

These were used as the control inputs for the standard PD equation to generate the control signal.

$$Z_{control} = K_D * Z_D + K_P + Z_E \tag{2}$$

Where K_D and K_P are the derivate and proportional control gains respectively.

The control gains were tuned iteratively to optimize towards a system that hovers accurately at steady state with minimal overshoot. Achieving a fast rise time was not a priority for this particular system, as maintaining stability is more important than fast operation. With use of the simulation software, these control gains were found to be $K_D = 1.3$ and $K_P = 0.2$. The result of these control gains within the simulator can be seen in Figure 19.

The rise time was 1.67s, overshoot was 2.5% and the system maintained an effective steady state within the simulator. This resulting hovering algorithm was deemed successful for simulation purposes, and is used as the standard hovering method for testing and developing the rest of the algorithms within this thesis.

4.3 Flat Surface Algorithm Development

The simplest surface for the UAV to track is a flat surface, and thus was used for initial development. This flat surface was modeled using the method outlined in Section 3.4, which was then utilized within the simulator.

4.3.1 Sensor Placement

For the placement of the any sensors in the UAV setup, there are several factors to consider. These include centre of mass offset, horizontal spacing and angle.

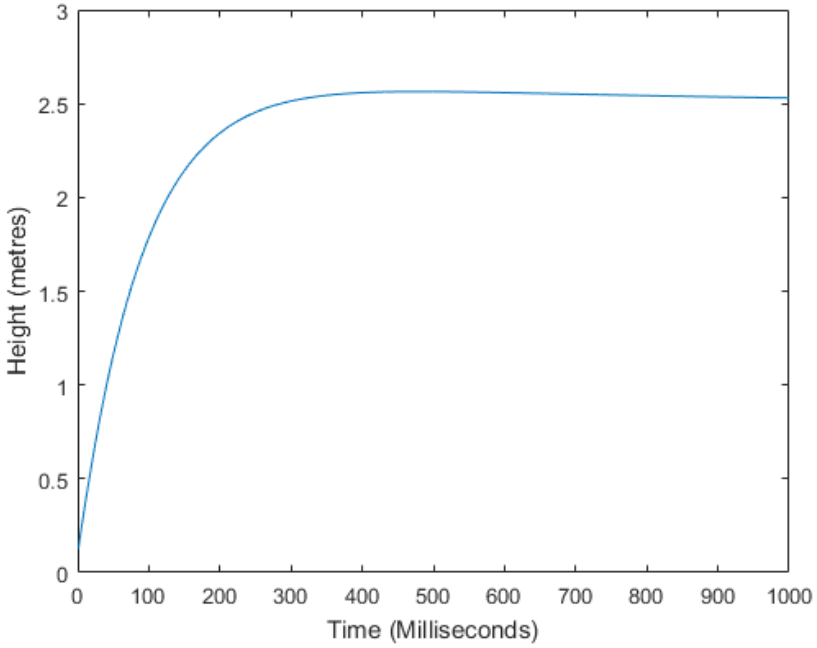


Figure 19: Height of quadcopter over time using hovering controller

These factors are extendable to any alternate sensor setups that may be implemented on a UAV, and are more apparent on larger systems with distant sensor spacing. This section will use two horizontal sensors as a reference example.

4.3.1.1 Centre of Mass Offset

On a physical UAV, depending on the design it can be difficult to place the sensors perfectly at the centre of rotation as there would be other physical objects present. The sensors must also be placed to provide a clear, unobstructed view of the world. This indicates that the sensor may need to be placed at an edge of the UAV.

This unavoidable offset can skew the sensor data slightly, and may be considered when designing algorithms. The further away from the centre of rotation a sensor is the larger the effects of this skew are felt. Consider the examples in Figure 20 and Figure 21, where a height sensor S_1 has been offset along the x-axis.

If the system were to pitch or yaw there would be no difference between the sensor readings shown in Figure 20, and would compare identical to the virtual sensor based directly at the centre of rotation. However, if the system were to yaw in either direction the offset sensor would physically move closer or further to the vertical surface as seen in Figure 21. If trigonometry is applied to this based on the angle of the UAV, it can be seen that the offset sensor returns

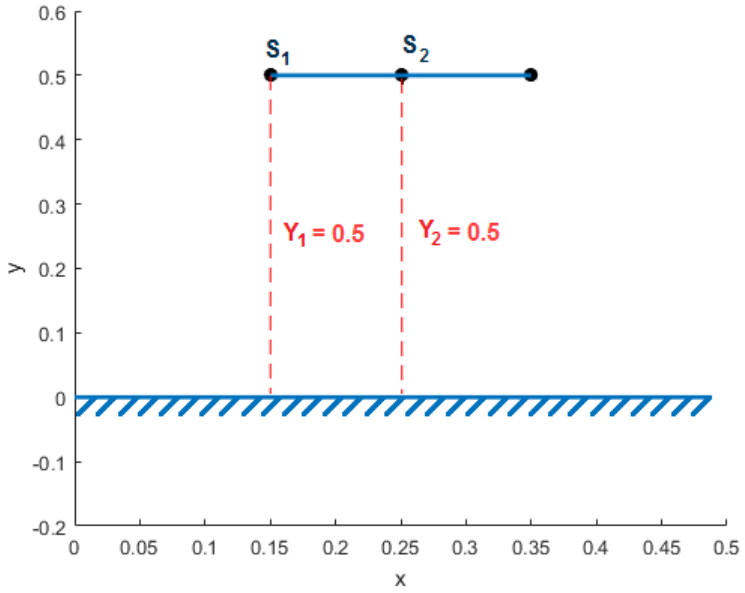


Figure 20: Parallel quadcopter system with two sensors ranging a vertical surface

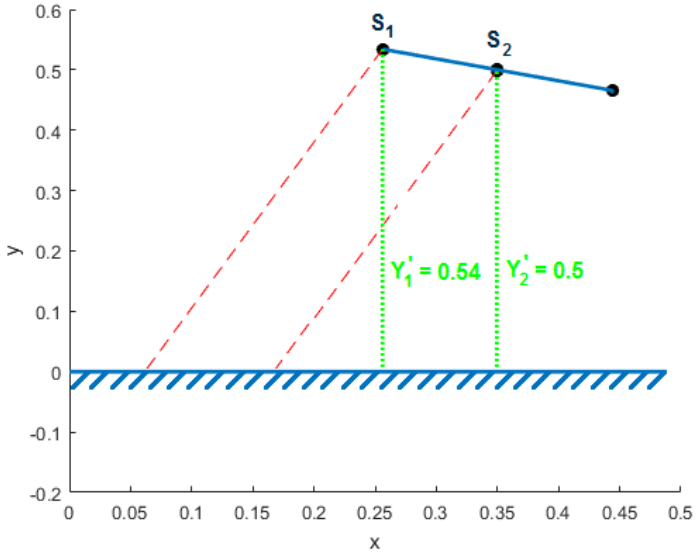


Figure 21: Angled quadcopter system with two sensors ranging a vertical surface

the wrong height value. This introduces a small amount of error that must be considered if the sensor placement has a large enough offset. Depending on the setup of the UAV system and the desired application this error could potentially be ignored. Having the two sensors in mirrored positions could also be used determine the actual distances from a surface by averaging their respective measurements.

4.3.1.2 Spacing

It is likely for multiple sensors in conjunction to be used for control purposes. The spacing between these sensors is a significant factor that can directly affect the results of control algorithms. A particular application of this could be adjusting the yaw of an UAV in close proximity to a flat surface. If the two sensors are further apart, then yaw adjustment can be far more accurate.

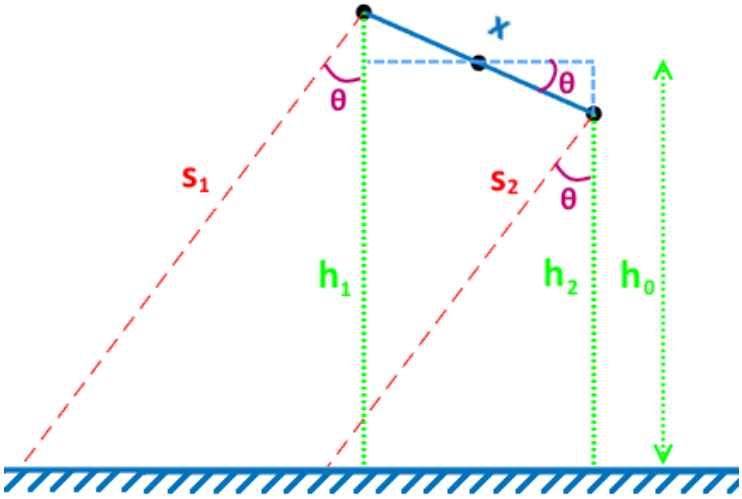


Figure 22: Angled quadcopter system with 2 locally spaced apart sensors ranging a vertical surface

Given the height of the UAV CoM (h_0), the current UAV angle (θ) and the spacing between the two sensors (x) the potential difference was derived. These parameters can be seen visually in Figure 22

$$\begin{aligned} S_1 &= \frac{h_0 + sin(\theta)(\frac{x}{2})}{cos(\theta)} \\ S_2 &= \frac{h_0 - sin(\theta)(\frac{x}{2})}{cos(\theta)} \\ d = s_1 - s_2 &= \frac{2sin(\theta)(\frac{x}{2})}{cos(\theta)} \end{aligned} \tag{3}$$

As can be seen by Equation 3, if the spacing between the sensors (x) is to increase then difference measured between the two sensors (d) also increases. This implies a further spacing gives a greater accuracy towards measuring yaw displacement from the difference between the two sensors. There is a trade-off in that it is difficult to physically mount the sensors far apart without specialized design, which in turn can also increase the bounding size of the UAV. For smaller spacings, these minor discrepancies can potentially be ignored as the difference would be relatively small and could be passed off as sensor error

4.3.1.3 Angles

The sensors mounted on the UAV can be angled outwards depending on the desired functionality. Angling the sensors outward can potentially provide a more accurate system with minimal drawbacks as long as it is implemented correctly. For a flat surface the key benefits are increasing the sensor detection spacing without physically mounting the sensors further apart. This can be achieved by knowing prior the mounting angle of the sensor and applying trigonometric equations to find the perpendicular distance from the surface to the UAV. For a UAV facing perpendicular to a flat surface this is simply Equation 4.

$$cos(\theta) * H = A \tag{4}$$

This angling method must be used carefully however due to the issue of specular reflection. These drawbacks are not related to any particular sensor model or product, but instead are the inherent result of the fundamental principles of the technology [24]. Using ultrasonic sensors as an example, if there is a large enough incident angle to a surface, not all of the wave is reflected back to the receiving sensor [64]. The type of surface is also relevant, and the smoother the surface less of the wave will be reflected back towards the receiver. The effect of specular reflection can be seen in Figure 23.

Specular reflection can vary widely depending on the type of sensor, the angle and the reflecting surface. For example, although laser range finders are less susceptible to specular reflection in general use cases, they have difficulties when ranging angled white or shiny surfaces [65]. In regarding to mounting angles for a UAV, it is recommended to undergo experimental tests of the intended setup to determine the maximum sensor angle before readings become inconsistent. For initial simulation purposes, the effects of specular reflection is ignored and any angle can be used.

4.3.1.4 Selected Configuration for Algorithm Testing on Flat Surface

For the initial flat surface tests a simple configuration was desired to be used. This is formulated as two sensors placed at a distance of 0.1m apart from the

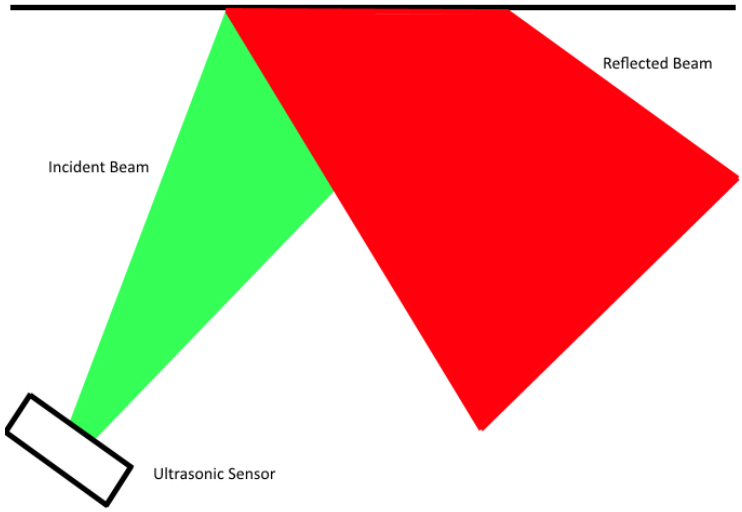


Figure 23: Visual Example of Specular Reflection

centre of mass and orientation in the same perpendicular direction as the UAV. This perpendicular configuration can be seen below in Figure 24, with the two blue arrows vectors indicating sensor direction with the base of the vector indicating the placement.

4.3.2 Vertical Surface Alignment through Yaw Control

The proposed solution assumes you are always near a vertical surface, and so following these assumptions allows a basis to be formed for initial wall alignment algorithms using proximity sensors. The purpose of wall alignment is to ensure that the UAV system is always facing directly towards the vertical surface, with adjustments being made as necessary. The system utilizes the perpendicular configuration of two proximity sensors, and uses PD algorithms similarly to the hovering algorithms seen previously in Section 4.2.

$$\begin{aligned} psi_{E\_PREV} - psi_E \\ psi_E = S_1 - S_2 \\ psi_D = \frac{psi_E - psi_{E\_PREV}}{dt} \end{aligned} \tag{5}$$

Where psi_E is the error between the two sensor distance inputs S_1 and S_2, psi_{E_PREV} is the error of the previous iteration, psi_D is the derivate error based on the change in error and dt is the simulation time step.

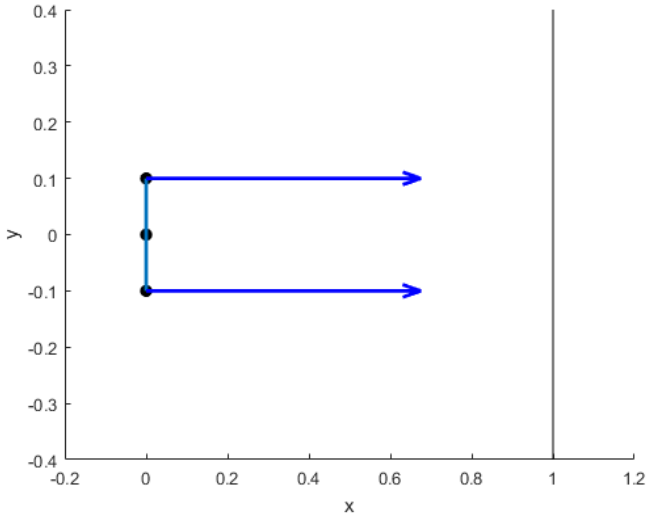


Figure 24: UAV in parallel sensor configuration

From this the resulting control signals are used within a standard PD equation to generate the output control signal.

$$psi_{control} = K_P * psi_E + K_D * psi_D \tag{6}$$

The control signal is fed through to the UAV to control the yaw.

The resulting output from the PD calculation forms the control signal that is then used to determine the rotor speeds. The control gains were determined experimentally within the simulation and were determined to be a K_P of 1450 and a K_D of 2500. The resulting control gains were also limited to be within a range of 2000 to -2000, to prevent an excessively large control signal being used when the yaw angle is too high. Figure 25 shows the control gains and rotation angle over time as the simulated UAV attempts to adjust its current orientation to zero. A bird's eye view of the simulation can be seen in Figure 26, showing the UAV rotating towards the flat surface. The yellow arrows represent the current orientation of the UAV, along the local x-axis.

The sensors view of the world can be seen in Figure 27, and show the two sensor values coming closer together over time as the yaw in adjusted. This sensor view is what informs the control algorithm calculations.

4.3.3 Maintaining Vertical Surface Distance through Pitch Control

A wall distance algorithm was implemented to maintain a set distance from a flat walled surface. This simplistic algorithm incorporated a simple distance average between the two proximity sensors in perpendicular orientation. This

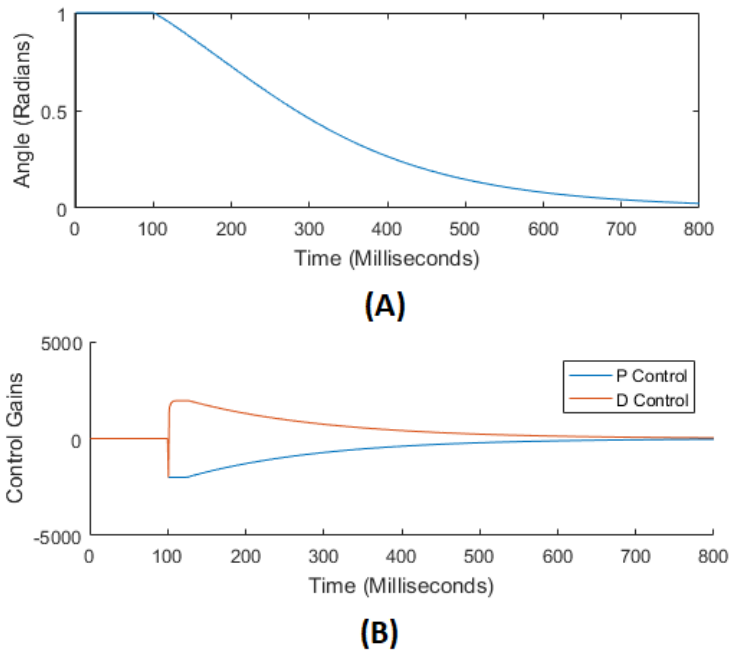


Figure 25: Adjusting yaw over time tracking a flat surface (A) Shows yaw angle psi over time (B) shows control gains over time

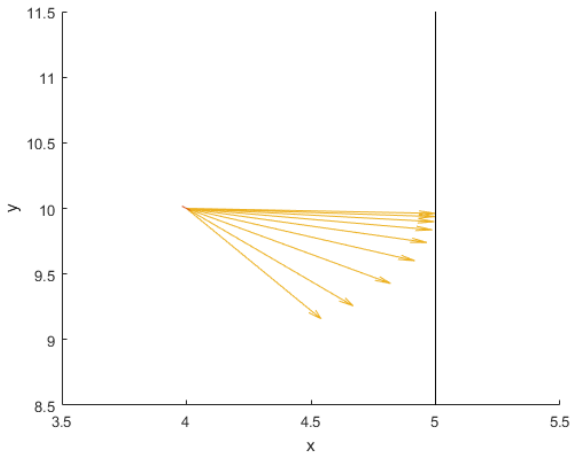


Figure 26: Birds eye view of simulated UAV adjusting yaw over time. Yellow arrows indicate the current local orientation of the system

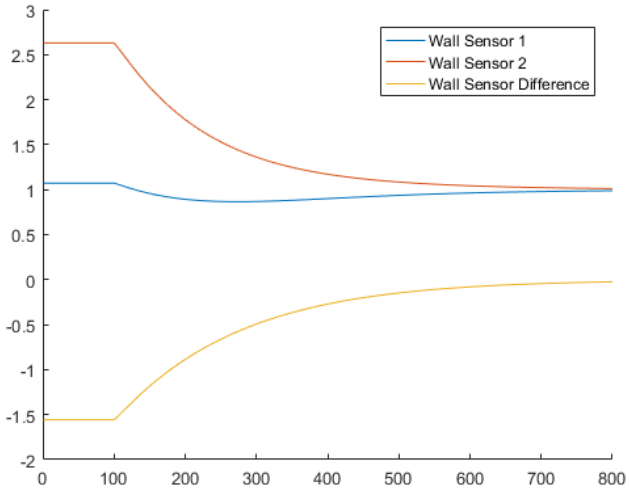


Figure 27: View from onboard sensors of quadcopter over time

follows similar to prior Section 4.3.2, but uses the average of the two sensors instead of the difference between them.

$$\begin{aligned} x_{E_PREV} &= x_E \\ S_{AVG} &= \frac{S_1 + S_2}{2} \\ x_E &= S_{avg} - x_{desired} \\ x_D &= \frac{x_E - x_{E_PREV}}{dt} \end{aligned} \tag{7}$$

Once again standard PD equation are used to generate the control signal to command the UAV

$$\ddot{x}_{commanded} = K_P * x_E + K_D * x_D \tag{8}$$

The produced outputs are the command accelerations which are then used to determine the pitch angle of the UAV. The control gains were determined experimentally within the simulation and resulted in a K_P of 3.1 and a K_D of 3.9. The system response with these control gains can be seen in Figure 28, where the system attempts to pitch the UAV from the x-axis position of 3 to 1.5. The result of this can also be seen successfully verified in the simulator within Figure 29. This figures shows the system launching to the desired height of 2m before pitching backwards to the distance of 3.5m from the surface. The orange arrows indicate the current orientation of the UAV in the local z-axis.

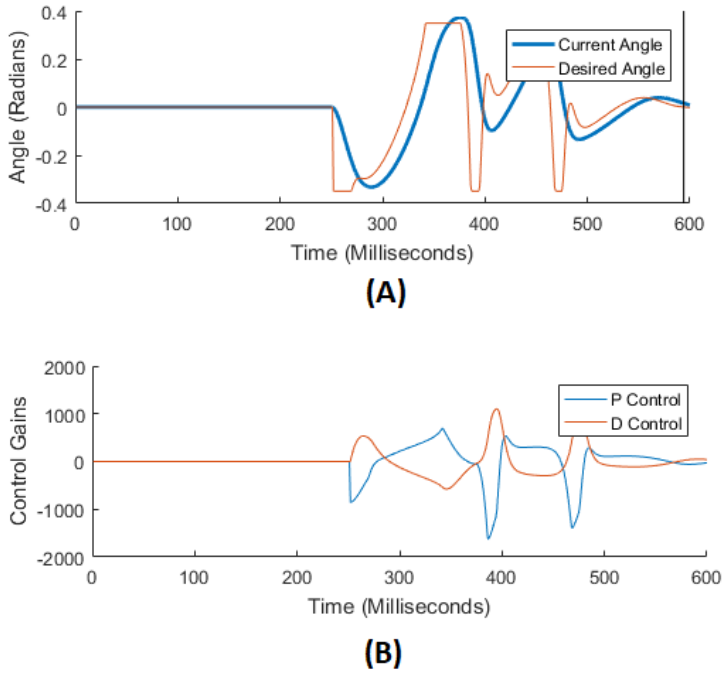


Figure 28: Adjusting pitch over time tracking a flat surface (A) Shows pitch angle theta over time (B) shows control gains over time

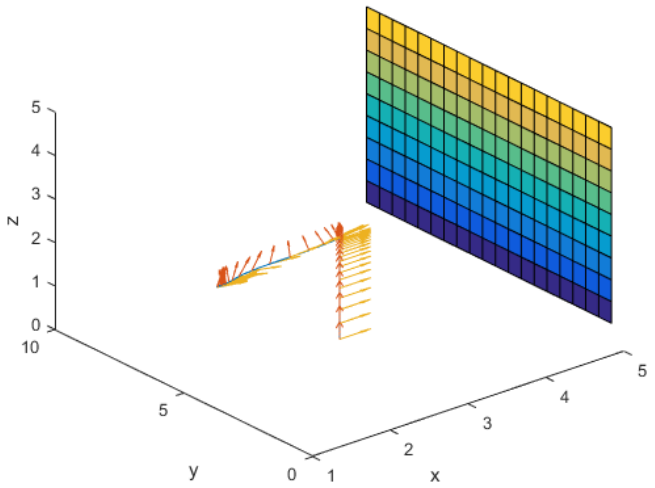


Figure 29: View of simulated UAV adjusting pitch over time. Orange arrows indicate the current z-axis local orientation of the system

4.3.4 Parallel Movement Alongside Vertical Surface with Roll Control

It is desired for the system to track parallel alongside a walled surface so that it can autonomously explore. In respect to primarily using proximity sensors for tracking, this is difficult for a flat surface. If the UAV were to move parallel to the surface, the on-board proximity sensors would detect no change in their readings. This effectively renders them useless for a flat surface and thus no basic control algorithms can be implemented for the roll angle. Alternate options were briefly investigated due to this fact.

The most straight forward approach would be using open loop control, tilting the UAV to move the system in a desired direction. This can be achieved by monitoring an on-board gyro to determine the current angle of the system and adjust accordingly. This approach is undesirable however, as the system can easily fly out of control without any feedback to correct the flight path or speed – even if the pitch and yaw are held steady. From testing within the simulator, setting the roll angle to a small fixed angle shows the UAV slowly accelerating over time to undesirable speeds. This can be seen verified in Figure 30, with the acceleration being indicated by the increasing spacing between the yellow arrows which indicating the UAV direction at a set time step.

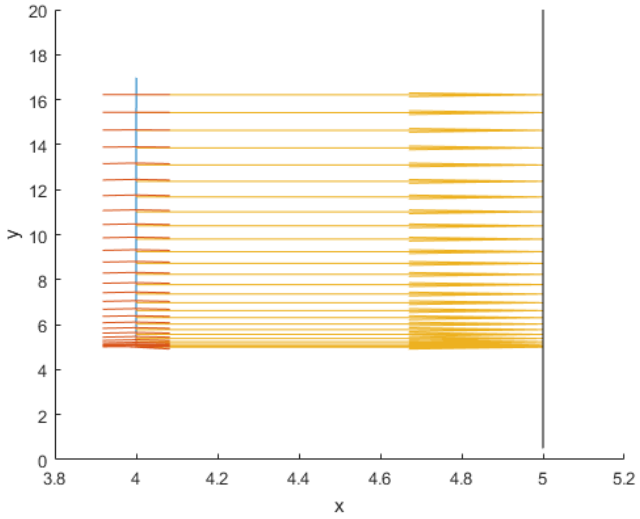


Figure 30: Birds eye view of simulated UAV adjusting roll over time.

The key issue with horizontal movement utilizing the UAV tilt angle that must be solved is determining the velocity of the system. It is desired to move at a fixed velocity as the UAV follows a surface, such that it does not stay stationary or accelerate to excessive speeds. If the velocity was known control algorithms could be used to adjust the system speed, however this estimating velocity is one of the major challenges currently faced for quadrotor systems [66]

Realistically, the optimal method for parallel movement alongside flat surface is to make use of sensor fusion. This method fuses together multiple sensor readings to give a more accurate localization of the system. Further localization methods would be required for the implementation of this such as GPS or wireless indoor positioning module. Delving towards these was outside of the project scope, so they will not be explored further. Development towards such methods can be seen explored by Tailanian Et Al, Azfar Et Al, Chan Et Al and Zheng Et Al [67–70].

4.4 Profiled Surface Algorithm Development

4.4.1 Sensor Placement

4.4.1.1 Spacing and Centre of Mass Offset

The spacing and centre of mass offset has no further major considerations to be made when moving to tracking a profiled surface. Simply prior sections 4.3.1.2 and 4.3.1.1 can be referred to.

4.4.1.2 Angles

Angling the sensors for sensing a profiled surface has further benefits and considerations that must be made, building upon Section 4.3.1.3. The key benefit of angling sensors for profiled surfaces is that it allows the system to 'look ahead' and observe surfaces that are approaching. This can allow for adjusting the position of the UAV prior to reaching a new surface profile.

These positives that stem from varied angles also come with their own trade-offs. The first main problem identified is that the sensor cannot see around corners, providing a slightly incorrect view of the upcoming profiled surface. The greater the angle of the sensor, the less the system can view around corners. This can be seen in Figure 31, with a virtual sensor angled forward detecting a profiled surface as the simulated UAV moves horizontally.

The actual view of the sensors is collated and can be seen in Figure 32. This shows the sensor measurements overtime where it can be seen that they do not directly match the prior wall profile in Figure 32.

The second major consideration that must be made when determining angle is the detection of upcoming surfaces that could potentially cause a collision. If the surface that the quadrotor is tracking sharply rises, the larger the angle used the sooner this can be accounted for. If a frontwards facing sensor detects the changing surface too late, it would not provide enough time to manoeuvre the system before colliding with the surface.

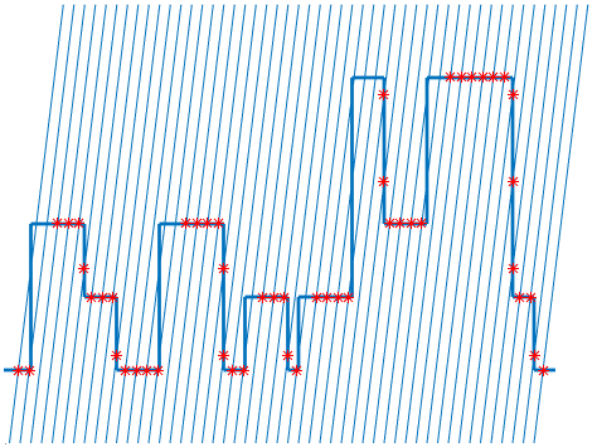


Figure 31: Angled sensor moving parallel to profiled vertical surface of time, showing intersections.

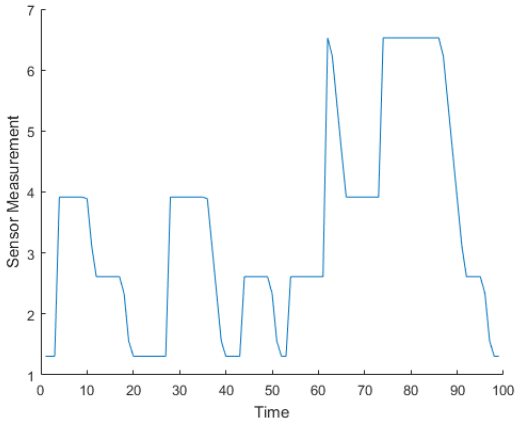


Figure 32: Angled sensor measurements moving parallel to a profiled surface over time.

4.4.1.3 Selected Configuration for Algorithm Testing on varied Surface

For varying surfaces, the same perpendicular sensor configuration cannot used to achieve the desired results in all cases. As discussed previously, without angling the sensors it is impossible to detect a rapidly increasing profile in quadrotor's upcoming path. Thus a second sensor configuration formed to handle more varied surfaces. The spacing was kept the same at a distance of 0.1m from the centre of mass, but one of the sensors was angled 20 degrees outward. This angled configuration can be seen in Figure 33.

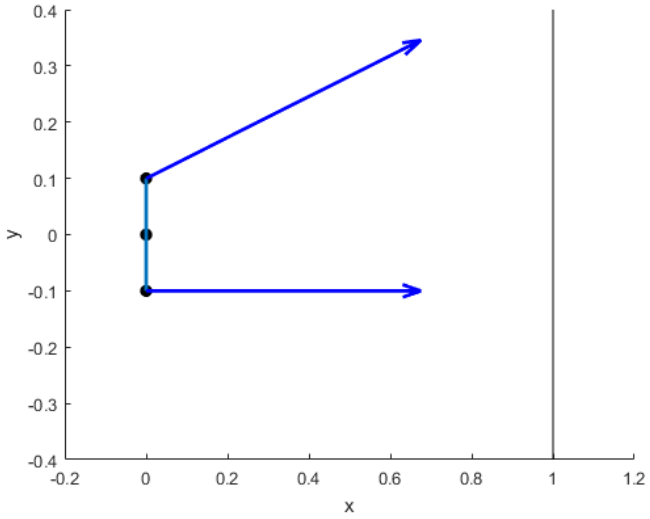


Figure 33: UAV in hybrid sensor configuration.

4.4.2 Vertical Surface Alignment through Yaw Control

Although highly useful for flat surfaces, controlling the yaw via proximity sensors becomes more difficult when the surface has a varying profile. Thus the prior wall alignment method discussed in Section 4.3.2 may not translate directly depending on the target surface. One of the key areas of focus for this thesis is achieving results with basic sensors and algorithms, so the following use cases are presented in which it the flat surface yaw algorithm could be adapted to work for particular varied surfaces.

Consider a smooth slowly varying surface that is desired to be tracked. As the surface is slowly changing, the UAV can continually adapt to keep it facing the direction of this surface. This can be seen in Figure 34, in which the virtual UAV tracks a sinusoid surface using two proximity sensors for yaw control in perpendicular configuration. In this figure, the pitch is adjusted to hold at a steady perpendicular distance from the surface and the roll is controlled such that the system moves at a constant velocity. This works effectively, which shows that more simplistic algorithms can be incorporated as long as the environment is known to be appropriate.

Angled configuration of the perpendicular sensors could allow for the system to look ahead and turn corners in the environment, with minor tweaks of the flat surface yaw algorithm. As it approaches a corner the look-ahead sensor can detect this and slowly turn the system towards the new surface. This approach was tested within the simulator for a corner surface using the flat surface yaw algorithm. Adjustments were made to the forward sensor reading using trigonometry to determine the perpendicular distance from the surface. Results of this showed that the quadrotor could turn the corner effectively, provided a slow enough roll speed to complete the turn before crashing into a surface.

Difficulties arise with use of the flat surface yaw algorithm when encountering rapid sharp changes in the profiled surface. If the surface profile was to continuously rapidly change the system would attempt to constantly re-adjust producing oscillatory behavior. This can be limited by capping the maximum yaw speed, but does not solve the core of the issue. This effect could lead to the system becoming unstable and in the the worst case losing control and crashing.

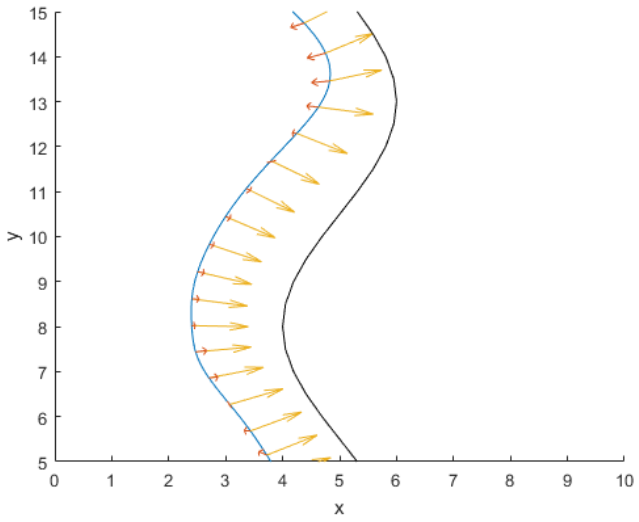


Figure 34: Birds eye view of simulated UAV adjusting yaw over time. Yellow arrows indicate the current local orientation of the system

4.4.3 Maintaining vertical Surface Distances through Pitch Control

Similarly to yaw control, the pitch of the UAV can also be difficult to control for varied surfaces. For slowly changing surfaces, adjusting the pitch can be implemented similarly to a flat wall system and will track the surface effectively. However, the issues occur when approaching sharp inclines or declines. If the UAV has not detected these in advance the system can crash into a sharp inclining surface. Conversely, if only looking ahead at upcoming surfaces the UAV system could pitch inwards too early and crash into a surface before it declines fully. An algorithm is proposed that that would solve these issues for this particular situation.

For the following proposed algorithm it assumes that the yaw is at a fixed perpendicular angle to the surface and the roll angle is tuned to track a fixed velocity. The two sensors are mounted in a hybrid configuration, with the first sensor angled forwards at a known angle (θ) and the second sensor mounted facing perpendicular to surface. This proposed algorithm attempts to map the upcoming surface based on the forward facing sensor in real-time.

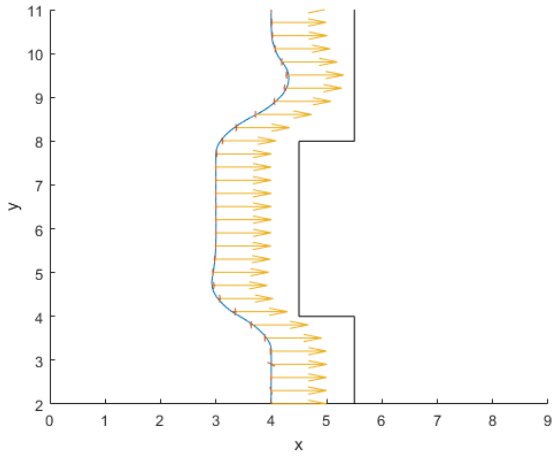


Figure 35: Simulation of UAV tracking a profiled vertical surface using a lookahead algorithm

This is achieved by forming a real-time array with contains a height map of the surface and the length of time until the system would reach a particular height. Each element of the array contains a distance measure from the tracking surface, and the corresponding indices indicate the estimated amount of time until that measurement is reached. The array is initially generated to contain all zeros, indicating that no data is present. The length of the array is determined by the update speed (dt) of the algorithm multiplied by the length of time in the future that the system wishes to measure(FDA_t). This array is known as the Future Distance Array (FDA) can be seen initially formed in Equation 9.

$$\begin{aligned} n &= dt * FDA_t \\ FDA &= [0_1 0_2 ... 0_{n-1} 0_n] \end{aligned} \tag{9}$$

After this base array is formed, the algorithm iterates through the following steps.

As the angle of the forward facing sensor is known, it can be split into two components: S_1^y and S_1^x. Distance indicates the perpendicular distance from the UAV to the surface (S_1^y), and the length indicates how far in the future that distance measurement was read.

$$\begin{aligned} S_1^y &= sin(\theta) * S_1 \\ S_1^x &= sin(\theta) * S_1 \end{aligned} \tag{10}$$

S_1^y is used to help determine which particular index to store the measured S_1^x. This is attained by dividing the S_1^y by the current horizontal speed of the system to find the time in seconds until the S_1^x measurement is reached. Finally adjusting this result by the time step, rounding down to ensure a valid integer index and then limiting the result to be within the array range provides the FDA array element to insert S_1^x into.

$$\begin{aligned} S_1^{TimeToDistance} &= \frac{S_1^y}{v_{current}} \\ FDA_i &= \left\lfloor S_1^{TimeToDistance} * \frac{1}{dt} \right\rfloor \\ 1 < FDA_i &< FDA_{length} \end{aligned} \tag{11}$$

However, S_1^xcannot be directly inserted into the equation, and the current measured distance (S_2) alongside the desired distance (xDesired) must be taken into account.

$$FDA(FHA_i) = S_2 + (xDesired - S_1Height) \tag{12}$$

Finally each iteration the each element in the FDA array is shifted one position to the left, which implemented the passing of time.

With the FDA algorithm updating correcting each iteration, it can then be used to track future distance measurements. Any time value in the future can be tracked, as long as it is less than the length of the FDA array. The index of that time value can then be checked, with the corresponding element showing the surface distance in that amount of time. This allows for the system to pitch towards that distance before the system physically reaches that point. In absence of a value in the sensor location (0), the system instead continues to use the last read distance measurement on the previous iteration.

The caveat that must be considered however, is the angle of forward facing sensor. Depending on the speed of the system and how far in the future the system is tracking, if the sensor angle is not large enough then mapping data may not be available. It is therefore recommended to dynamically shift downwards to the first viable measurement in the FDA array if there is no data past the selected future tracking time.

The algorithm presented was tested within the simulator to verify the functionality. This was completed using a FDA array length of 10 seconds (1000 Elements), a future tracking time of 1.5s, a forward sensor angle of 26.5 degrees, a time step of 0.01s (dt) and a horizontal velocity of 0.5m/s ($v_{current}$). The result of this can be seen in Figure 35. This shows the system successfully pitching outwards to avoid an upcoming profiled surface, followed by the system correctly pitching inwards at the correct time.

4.4.4 Parallel Movement alongside Vertical Surface through Roll Control

Horizontal movement through roll control has difficulties in implementation when using only proximity sensors. As discussed previously in Section 4.3.4, for a flat surface it is unfeasible to determine any reliable data for control. However, if the surface has a varying profile there is resulting data from the proximity sensors that can be potentially used.

If alternate primary method is used to determine the horizontal velocity of the system in real-time, proximity sensors in an angled configuration can be used in conjunction to help inform the desired speed. Consider the UAV system approaching an inside corner. The desired result is for the horizontal speed of the UAV system to slow down such that it can navigate the corner accurately without crashing into the surface. Having a fixed roll speed is difficult for this situation, and so an adaptable algorithm is presented.

Proximity sensors in an angled configuration can be effectively used in this situation to inform the system that it needs to slow down and thus adjust roll control accordingly. The forward facing sensor can detect a sharp approaching incline and subsequently slow down the UAV system to account for this.

A mapping algorithm can be used for such a case, which monitors the frontward facing angular sensor's error and maps this value to a new desired horizontal speed. This in turn adjusts the roll of the UAV which slows the horizontal speed to the desired amount. A basic mapping formulae for this purpose can be seen in Equation 13.

$$x' = \frac{(x - in_{min}) * (out_{max} - out_{min})}{(in_{max} - in_{min}) + out_{min}} \tag{13}$$

This equation can be simplified for the particular situation outlined by replacing the following variables

$$\begin{aligned} in_{min} &= 0 \\ in_{max} &= E_{max} \\ out_{min} &= v_{max} \\ out_{max} &= 0 \end{aligned} \tag{14}$$

in_{min} can be set to zero which designates that the sensor is tracking the surface accurately with no deviations and therefore can be mapped to v_{max}, the maximum horizontal velocity desired. in_{max} can be replaced with E_{max}, indicating the maximum desired error which when reached sets the output horizontal speed to 0, or out_{max}. The simplified equation can be seen as follows.

$$x' = \frac{x * -v_{max}}{E_{max} + v_{max}} \tag{15}$$

The resulting speed must also be limited, such that it does not exceed the range between v_{min} or v_{max} and this is completed in one last final step.

This method was implemented into the simulation for experimentation using the angular sensor configuration. v_{max} was set to 0.5m/s and E_{max} was also set to 0.5m. The yaw is set to a stay fixed perpendicular to the horizontal surface. The pitch is controlled based on the reading from the front facing sensor which allows the UAV to navigate around approaching contours rather than coming to a halt, based on prior Section 4.4.3. The results of this simulation can be seen in Figure 36, where the UAV simulation successfully navigates an inside corner. Comparatively, the results of maintaining a fixed velocity can be seen in Figure 37 where it is shown that without this dynamic velocity method the system would crash into the surface.

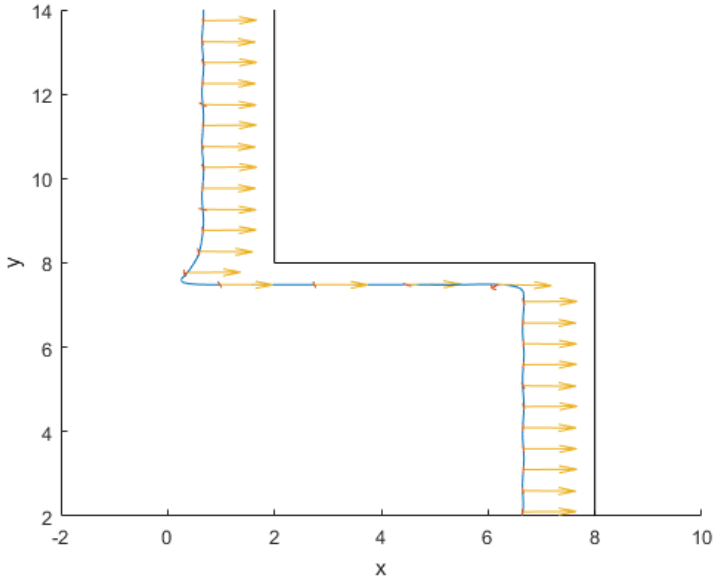


Figure 36: Simulation of UAV tracking a inside corner surface with roll adjustment control

4.5 Summary

This chapter explored a variety of different options for autonomous control algorithms using proximity sensors, with a focus towards isolated control channel. The built simulation was used as a tool to experiment with and verify these potential options.

Considerations of sensor placement and angling onto a UAV were presented for both flat and varied surfaces. The investigation presented shows thats for small changes in placement or angle resulting data can be used directly, ignoring the

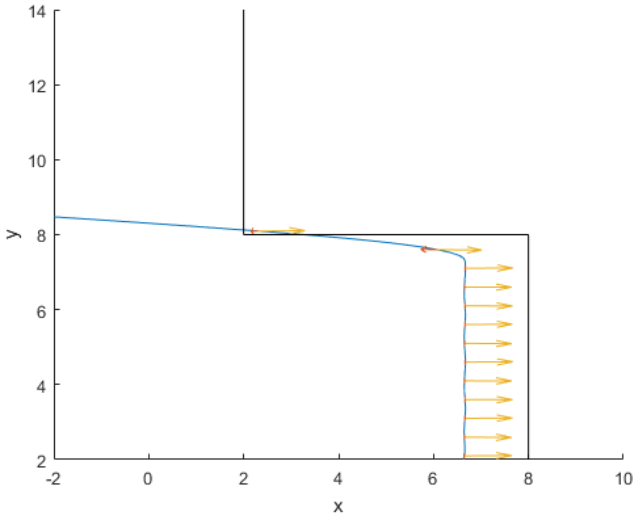


Figure 37: Simulation of UAV tracking a inside corner surface without roll adjustment control

minor discrepancies. However, for large angles or offsets in sensor placement the skewed resulting data should be factored into account and adjusted accordingly.

Basic control algorithms for isolated control channels are outlined for flat surfaces. These provided effective control options for both pitch and yaw, with successful results within the simulation. However, for roll control it was shown that for flat surfaces there is insufficient data to use solely proximity sensors for direct control. This leads to the conclusion that potentially a fully autonomous system cannot be fully achieved solely relying on proximity sensors, and integration with proven methods could be prudent for future work.

Options for tracking algorithms for profiled surfaces are also presented. These once again were developed for isolated control channels and focused on specific use cases that could potentially be adapted to a more generic approach. With use of yaw control it was shown how basic algorithms for flat surfaces can be adapted to work for varied surfaces, but with limitations on what surface profiles they could be effective for. Pitch and roll control algorithms showed the possibility of look-ahead sensors for varied surfaces to ensure the system maintains a safe operating distance at all times.

5 System Design and Validation

A physical system was desired to be built so that control algorithms that have been explored could be experimented with on a real-world system. This would allow for the validation of such algorithms to confirm their functionality. This section presents the efforts towards achieving this goal of creating such a system.

This chapter describes the designed quadrotor platform and initial function testing. First the requirements of the system are presented and an overview of the flight platform hardware is given. Details on the electronics implementation is then fully described, forming the basis for the entire system. Information is provided in regards to the software written to control the quadrotor. The entire final system is validated to confirm functionality, and experimentally tested using isolated control channel algorithms.

5.1 Requirements

Before the system was designed fully, the desired requirements were formulated. These provide a basis for future design choices and component selections. The core requirements are listed as follows

1. Designed system hardware and software should be adaptable to any UAV system

Although a singular system will be constructed for the purposes of this thesis, it is desired that the created method is extendable to alternate UAV systems. This is important as this is an evolving, unexplored area of design space that could easily require major system changes in the future. By keeping implemented designs modular this allows to for easy experimentation on different types or sizes of UAVs.

2. Require the ability to isolate and autonomously control a singular channel at any given time

For initial testing purposes of control algorithms using proximity sensors, it is useful to be able to control a singular channel at any given time. This allows for isolated testing of each of the UAVs control channels, confirming their functionality. This also enables the possibility of hybrid or assisted control for quadrotor flight operators.

3. Require the ability to swap between manual and autonomous control, for safety and usability.

It is desired to remotely be able to swap control channels between autonomous and manual control states. This is useful for several reasons, the first of which are safety and cost aversion. While testing a UAV system, if unexpected errors are to occur then the system could crash damaging valuable components or create dangerous situation for bystanders. Having this remote safeguard allows for

an operator to regain control of the system and safety land. Another useful reason is for testing purposes. When testing various algorithms which make use of proximity sensors, there may not be autonomous protocols to fly the system to a desired starting location. Commencing a flight with full manual control then swapping a certain channel to be controlled autonomously mid-flight allows for more flexibility in the testing of various algorithms.

4. Require the ability to log relevant data such it can be reviewed for testing and tuning purposes

Logging various flight data through test flights is very important. Early on it can be used effectively for tuning a flight system. Later on it can be used to provide concrete results which could be used for validation purposes.

5. Require the software capability to read in sensor data and translate them to control outputs

The core of this thesis project is be able to generate control outputs based on proximity sensor inputs, therefore this requirement is necessary to reach the project goals.

5.2 Hardware

5.2.1 Flight Platform

Although the system being created is desired to be modular across multiple UAV platforms, a specific flight platform is still required for validation of the design. As the focus of this thesis is not on the UAV construction itself, an off-the-shelf UAV is selected that can be extended upon as necessary. The model chosen was the DYS 320 Full Carbon Fiber Folding Quadcopter. This is a small, lightweight quadrotor with a full carbon fibre frame, a preconfigured flight controllers and powerful 2300kv motors. This system also comes with an on-board CC3D flight controller running OpenPilot software to initially configure and control the quadrotor, as well as a FrSky X8R radio reciever. The system is powered by a small 12V LiPo battery which allows for approximately 15 minutes of flight time.

This flight platform can be seen modelled in Figure 38. Each individual part was measured and constructed digitally which were then assembled together create a full CAD model of the base system. For scale purposes, the length of the core chassis is 0.22m, the width is 0.055m. This is extended by the rotor arms at a length of 0.13m each.

5.2.2 Modifications

Extensions to the base flight platform are required so that space is available for further components and additions. To simplify the design of these extensions a 3D CAD model is utilized to virtually implement the mounting of the

Figure 38: Solidworks Model of Quadcopter

components before attempting this physically. The additions included a secondary lower mounting platform and mounting brackets for sensors. Using the virtual designs these extensions created from laser cut Perspex and attached to the main unit. This successfully allowed room for the microcontroller and all additional sensors. The final produced prototype can be seen in Figure 39.

5.3 Electronics

To enact the goals of this thesis project, additional electronic components are needed to meet the requirements listed in Section 5.1. A microcontroller is required to be selected to read in sensor data, enable autonomous control and log relevant results. Appropriate sensors also need to be selected that can effectively track a profiled surface. Custom circuit boards need to be produced to meet the autonomous control requirements listed previously

5.3.1 On-board Microcontroller

A microcontroller is required to read in sensor data and generate control outputs. For this project the controller selected was the simple Arduino Mega 2560.

The Arduino Mega 2560 is a versatile and compact development board released in 2010 [71]. It features a programmable ATmega2560 microcontroller running at a 16MHz clock speed. It is widely used across many different areas, including the development of UAVs where researchers have used the Arduino platform as a means to develop and test various prototypes in the past [72–74]. A large number of pin inputs and outputs are available including 54 Digital Pins and 16 Analog pins. This is beneficial as it having more than ample pins for development purposes means that there will never be any hardware limitations while

Figure 39: Quadcopter prototype system

prototyping the system. The other benefits of using the Arduino platform include inbuilt simplified programming functions, vast array of available libraries and low cost hardware.

5.3.2 Sensor Selection

Proximity sensors are required to provide feedback to the quad rotor control system which can subsequently enable autonomous control. These are to be used to the distance to a nearby surface. There are multiple options for sensor selection to choose from, including ultrasonic, infrared, sonar and laser based sensors.

The selected choice of sensor to use was a laser range finder. This has the advantages of having high accuracy, fast response time and convenience of use [75]. This is important for a UAV system relying on proximity sensors for control, as an airborne system must be able to quickly and accurately respond to measurements to avoid destabilizing or crashing the system. They have been also used in prior research for indoor GPS-denied navigation of UAVs to great effect [76,77].

A laser range finder uses a laser to measure distances. A laser beam is sent from the sensor towards a target and the time of flight is measured. Multiplying this by the speed of light provides the distance to the target. Using a laser range finder provides a highly directive, monochromatic source while maintaining minimal power consumption. These features are important to improve

signal to noise ratio, ensure the accuracy of the measurement and to elongate operation time. [25]

To determine the distance measurements from the quadrotor directly using proximity sensors, a LIDAR-Lite 2 laser rangefinder was selected [78]. This is a powerful laser based measurement solution which can measure up to a 40m range with 1cm resolution. It operates from a 5V power source while only drawing 100 milliamps at peak power, which is negligible in relation to the motor's drawn current. The repetition rate of the sensor can reach up to 500hz, which would allow for the quadrotor to quickly adjust to any changes in distance measurement. It can communicate the resulting measurements either over I2C or PWM channels.

5.3.3 Data Logger

Tracking flight data is highly important for tuning and the validation of results. One method of recording data is to use a data logger, which records relevant flight data of the system in real-time to be reviewed later.

The selected component to log data was the SparkFun OpenLog data logger. This open source data logger works over a simple serial connection and can support microSD cards up to 64GB. The board runs using an onboard ATmega328 with an average current draw of close to 5mA. It has a simple command interface and allows for configurable baud rates of up to 115200bps. A standard 32GB micro SD card was also sourced to use in conjunction with OpenLog system.

5.3.4 Modular Shield

To meet the requirements outlined by this thesis, custom hardware was designed. This would enable allowing the system to swap between manual and autonomous control, as well as the isolation of singular control channels. Additionally, further traces were added to simplify the wiring of selected components on the system. The modular printed circuit board (PCB) is designed in the form of a shield for the Arduino Mega 2560. This allows to easily connect all the external components to the Arduino Mega 2560 and minimize wire inductance by using traces instead. The created schematic can be seen in Appendix B.

To achieve the goal of swapping between manual and autonomous control, an alternating buffer system is designed. The PWM signals generated by the radio receiver and autonomously by the microcontroller are each propagated through a separate buffer. It is desired that at any given time one of these buffers will always be enabled and propagating PWM signals. To attain this, the control channel determining the operation mode is inverted between the buffers. Due to this, one of the buffers will always be enabled at any given time providing a PWM signal to the flight controller. A simplified diagram outlining this method can be seen in Figure 40

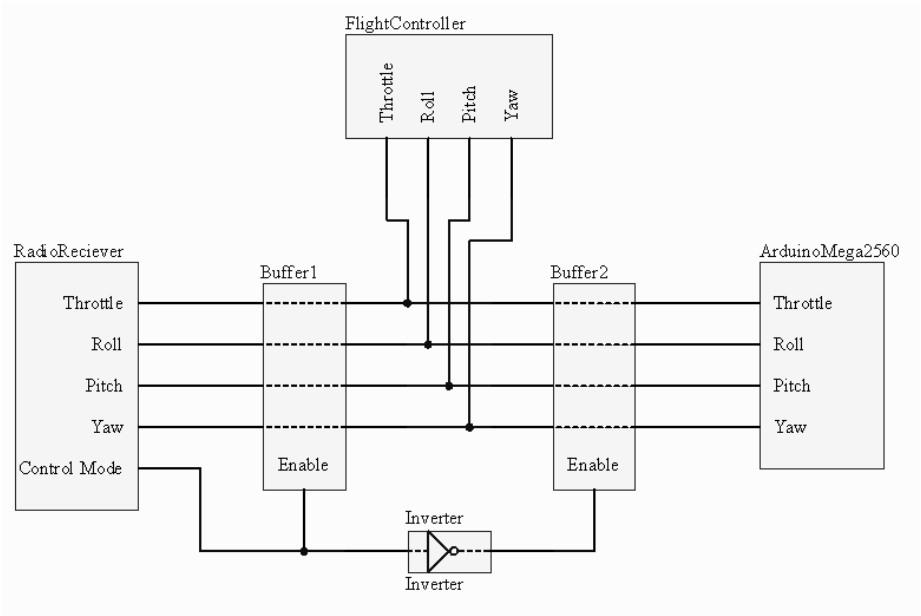


Figure 40: Simplified Diagram of Buffer Plan for controlling Quadcopter

The SN74HCT245N Transceiver was selected as the buffering component. The SN54HCT04 Inverters was used to invert the buffer control signal.

Headers for wired connections were made available so that there is the option to bypass the buffers and feed into the flight controller directly. This header selection allows a user to wire the radio signal outputs directly to the flight controller or through the buffers. This design choice allows for the isolation of particular control channels. By directly wiring all other control channels to the flight controller and disabling the autonomous outputs on the microcontroller this allows the hybrid control of the system. This hybrid control is utilized generally within this thesis as one channel controlled autonomously and the rest manually, but can be modified to any configuration necessary. Figure 41 shows the created PCB shield, indicating the ground and power headers, the header to bypass the buffer, the header input which can switch between control modes and the output header to the controller.

Further features were implemented onto the modular shield to simplify the connections between the microcontroller and the external components. A trace connected the Serial3 from the Ardunio Mega to a connection port where the logger can be plugged into. Another connection port was made for the LIDAR-lite 2 laser range finder which could be plugged into directly. Additionally extra power and ground ports were implemented as the Ardunio Mega has a limited number available. This allows for flexibility in implementing any further sensors or components that otherwise would not have an available power port. Finally multiple LEDs were added for debugging and feedback purposes.

The final PCB and schematic design can be seen in Appendices B and C.

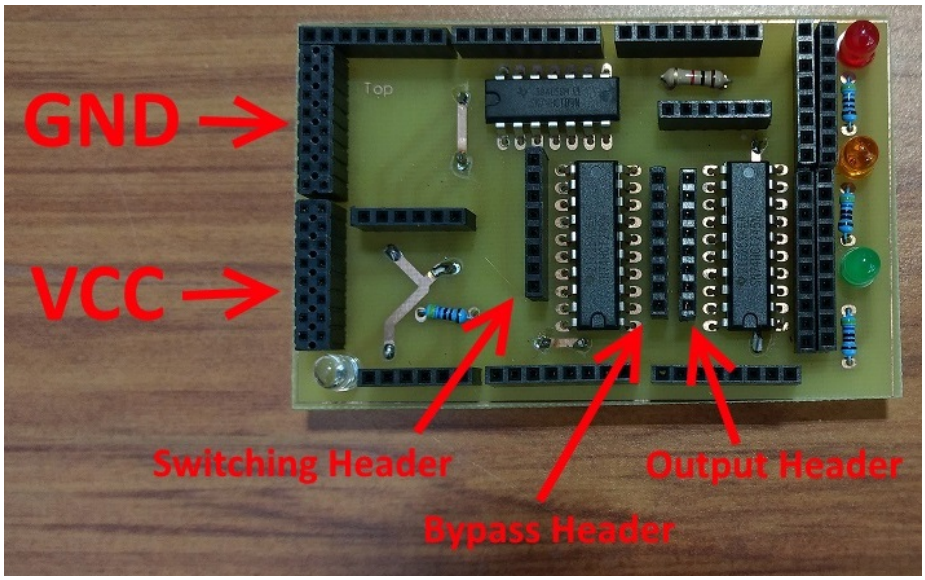


Figure 41: Prototype PCB Shield

5.3.5 Additional Operation Mode Microcontroller

The operation mode of the controller (Autonomous or Manual) is configured by the flight operator through their radio controller. Control of the current operation mode is then received by the radio receiver in the form of a PWM signal, unsuitable for direct control of the alternating buffers. Therefore, a secondary dedicated microcontroller is required to convert an incoming PWM signal to a binary output. This was separated from the main microcontroller to allow for focused operation of their distinctive tasks. The selected microcontroller was the small ATTINY85 development board.

5.4 Complete System

The complete system architecture can be seen summarized in Figure 42. The radio transmitter sends commands to the receiver, which in turn sends PWM control commands to the modular shield and the PWM operation mode to the secondary flight controller. The modular shield is mounted fully onto the microcontroller, which interfaces with the sensors, the data logger and the flight controller. The flight controller then drives to motor controllers to actuate the motors. The system is fully powered by an on-board battery, with most systems receiving power through the modular shield, aside from the motors who have a separate direct connection due to their high power consumption.

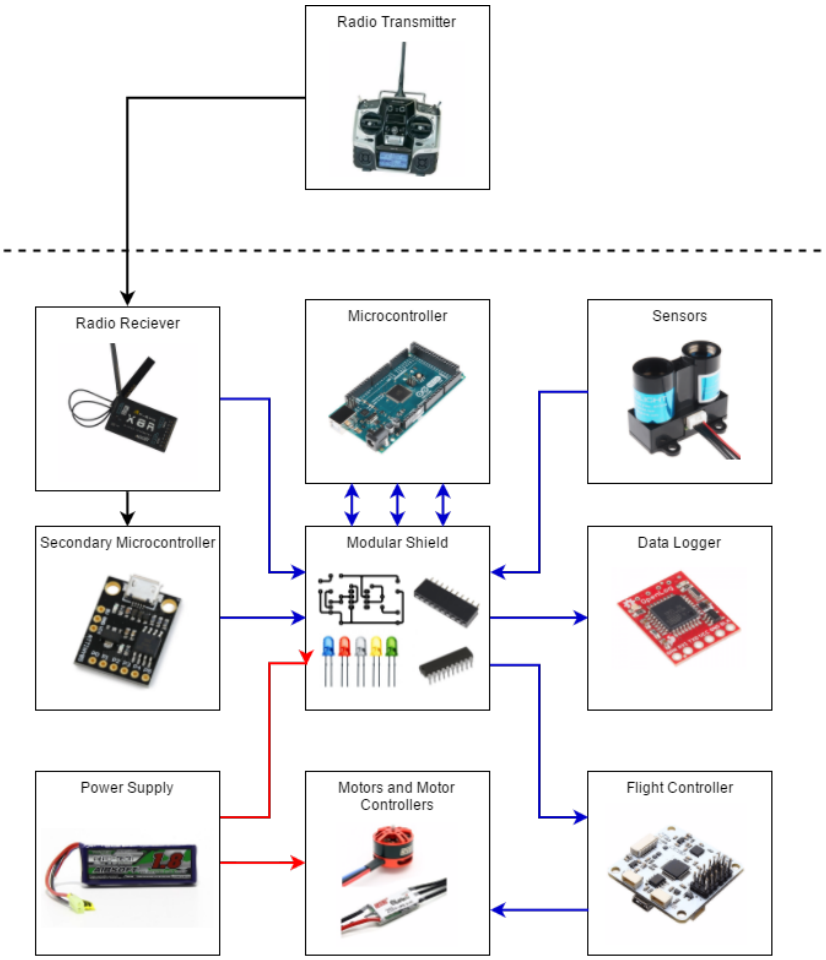


Figure 42: Quadcopter System Architecture

5.5 Software

Software for controlling the system was required to be written on the Arduino Mega 2560 Microcontroller. The embedded software was written entirely in C++ with use of Arduino libraries where necessary. Many of these libraries were custom written from scratch to allow for more flexibility in design and how they operate.

The software was written to meet the following goals.

- Emulate flight controller outputs
- Read in sensor values
- Generate control outputs based on sensor readings
- Log relevant data

An overview of the plan of the software architecture can be seen in Figure 43.

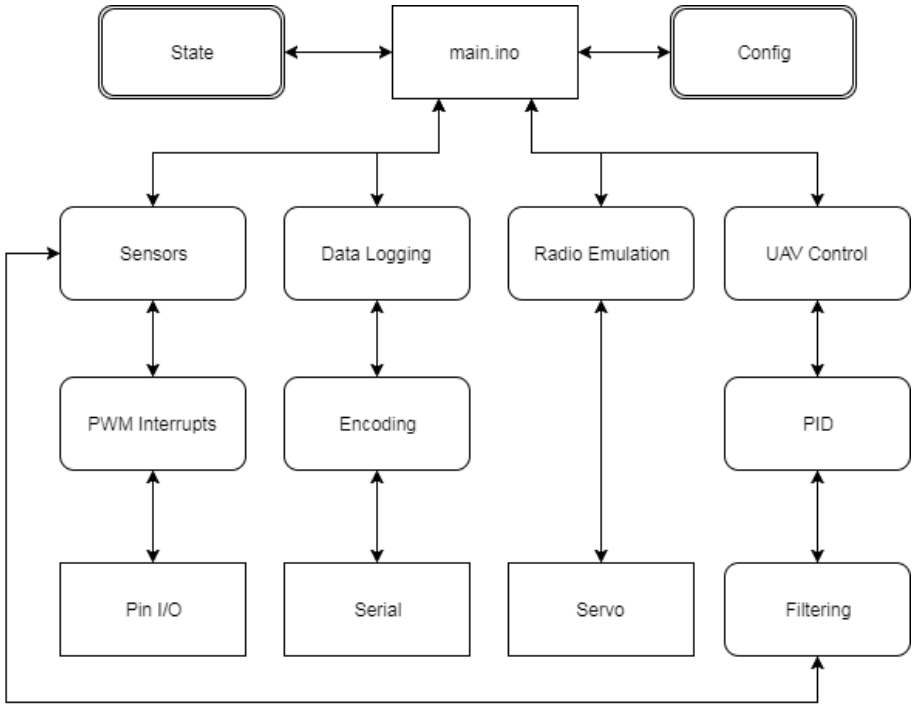


Figure 43: Simplified Diagram of Buffer Plan for controlling Quadcopter

The software was designed to be built in separate modules that run independently of each other so they could be modified or disabled as necessary. “Main” controls when each sub-function module will run, currently using basic round robin scheduling. The current state of the system is passed into each of these modules for use and a general configuration file is used to tune a variety of constants.

5.5.1 Main

The ”main” module controls the initialization and scheduling of all child modules within the program. It firstly reads a config file which set various constants through the program such as control gains, desired hovering distances and pin definitions. Following this it runs each of the child modules at a fixed rate, reading in sensor values, running control algorithms, emulating the radio signals and finally logging flight data. While the main function runs these child modules it passes through the current ”state” of the system through to each module, which update and return the state as necessary.

5.5.2 Sensors

The "sensors" module handles everything related to reading in the various sensor values. The sensor used is the LIDAR-2 Laser Range finder which communicates the measured signal using PWM. The sensor module contains functions to both initialize and update all sensors that are desired to be used, and can be extended to accompany further sensors if needed. The data that is read in from the pin inputs is stored in a global state function.

Underlying software libraries were written to parse these PWM input signals using interrupts, which enables multiple interrupt pins on the microcontroller to measure the PWM width. This is achieved through the detection of edge changes in the signal.

Filtering functionality was added to the system to minimize signal noise and sensor error. Simplistic methods were implemented in the form of rolling average filters and rolling median filters. These functions can be used to determine a current sensor reading where necessary as opposed to reading the value directly.

5.5.3 Data logging

The "data logging" module logs the operation of the system over time so that it can be reviewed at a later time. This is achieved by using serial communication logging to an external SD card. The module contains functions to first initialize the serial ports and subsequently log initial strings detailing various constants from the global config file. Following this, the module also contains functions to update and send real-time data across the serial port.

An encoding method was introduced to condense the information being sent to the SD card, to minimize the amount of data being sent each iteration therefore reducing overall CPU load time. A decoding script was written with Python on a PC to extract the relevant data that had be previously encoded by the microcontroller and stored onto a SD card.

5.5.4 Radio Emulation

The "Radio Emulation" module generates the signals being sent which in turn operate the motors via a secondary flight controller. The signals produced emulate the PWM signals sent from a manually operated radio transmitter. To achieve this, a PWM signal is generated using the Arduino Servo library and output through an I/O pin. The module contains both initializing functions to instantiate the pins and functions to update required control channels with the desired PWM signal. Further precautions were added in this module to ensure the PWM signal being sent is does not exceed the expected result on the flight controller.

5.5.5 Motor Control

The "Motor Control" module handles the algorithms used to generate control signals for each of the quadrotor control channels. This module can be adapted rapidly to test new algorithms for different control channels.
PID control is one of the most commonly used methods for simplicity and robustness [79]. As implemented algorithms are expected to use this method heavily a PID library is used to simplify implementation.

For validation and testing purposes of the UAV prototype initial control algorithms were written for the pitch adjustment using a singular laser range finder, allowing the system to track a vertical surface. Firstly an initialization function was created which defines the PID gains, the desired set point and the outputs limitations. An update function is then used following this continually updating the controller based on the measured sensor data.

5.5.6 Secondary Microcontroller - Operation Mode

Further software was written for the secondary microcontroller to convert a PWM input to a binary output. This would be then used to control the operation mode of the quadrotor. This was achieved by measuring the PWM width using interrupts, and then updating an output based on a threshold width.

5.6 Testing Results

5.6.1 Testing Environment

The prototype quadrotor system was tested in a controlled indoor environment. This was a 4m by 4.6m enclosed corner area, with a trackable wall on one of the edges. For safety, there was netting rigged allowing for viewing of the system in action while protecting bystanders from any danger. A picture of the testing environment can be seen in Figure 44.

5.6.2 Isolated Pitch Control Experimental Testing

With use of the testing environment, experiments were performed to further test the functionality of the system and the response of control algorithms. The quadrotor was setup to isolate the pitch control channel to be controlled autonomously through radio controller emulation. A singular laser range finder was configured to face forward on the quadrotor to be used for feedback control purposes.

The experiment was run with the operator launching the UAV manually and bringing the quadrotor to a steady hover state facing a vertical surface. Once in position, the operator then swapped the system to an autonomous mode where the sensors would take over and attempted to autonomously control the UAV.

Figure 44: Testing Environment for Prototype Quadrotor System

The pitch control algorithms were based off prior Section 4.3.3. Within the simulation this provided accurate autonomous control of the quadrotor for isolated pitch control. The control gains were selected to be a similar ratio but scaled appropriately to produce the correct emulated command signal which would be then converted to a PWM signal. After experimental testing, a suitable K_P was found to be 0.2 with a K_D of 0.25 based on measuring the most optimal control results. The wall tracking distance was set to 1.5m, which corresponds to a sensor reading of 1500.

Limitations were also placed onto the output PWM signal, to ensure the flight controller never receives a signal which is too large and tilts the UAV to an uncontrollable angle. The tracking results of the experimental testing can be seen in Figure 45.

The first noticeable result in Figure 45 is a constant steady state error that is present. The average distance held across the course of the autonomous mode tracking is 1612mm, 112mm from the desired result. Secondly, it is clear that there are distinct fluctuations throughout the course of the autonomous control. These fluctuations range from a minimum of 1551mm to a maximum of 1704mm. This indicates fluctuations $\pm$76.5mm from the average steady state position. The rise time of the initial tracking adjusts from the incorrect position of 1250mm to the average steady state position in approximately 2500ms. The steady state error and fluctuations can be explained through investigation of the output PD control gains, which can be seen in Figure 46.

From Figure 46 it is clear that the control system errors derive mainly from the derivative control of the system. Inspection of proportional control shows a

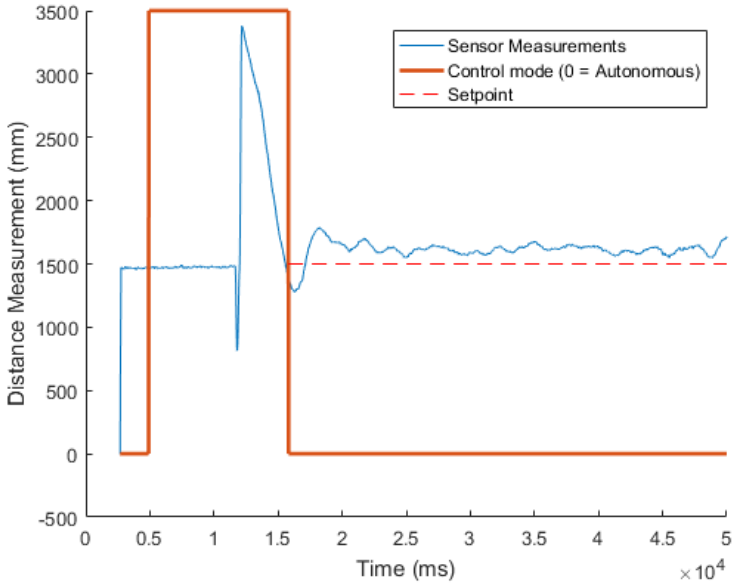


Figure 45: Tracking results of experimental testing

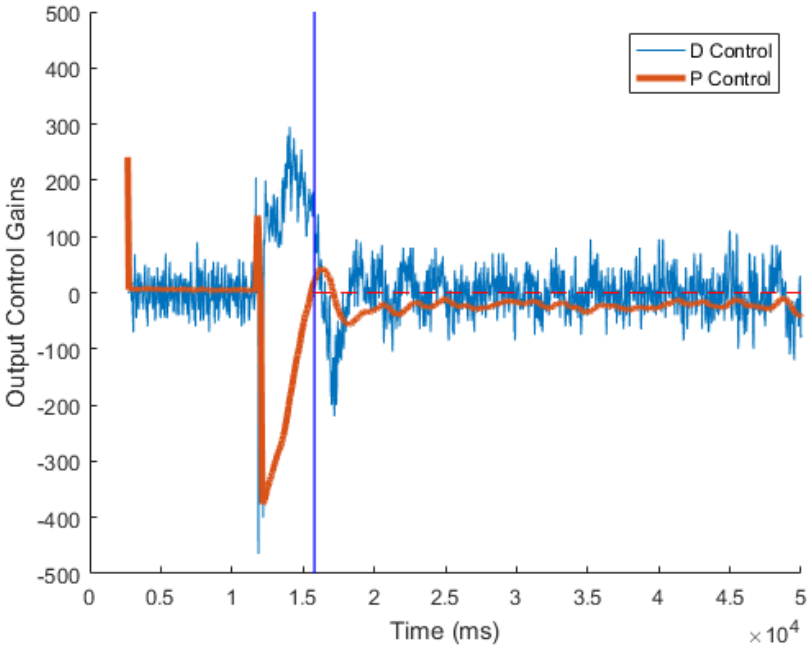


Figure 46: Output control gains of experimental testing

smooth control output and a constant offset as it attempts to rectify the steady state error present.

The derivate control noise is a by-product of the laser range finder sensor error. The derivative action amplifies the noise in the original signal which is then seen at the controller output. This can be seen visually by overlaying the sensor measurements with the output derivate control gains, provided in Figure 47.

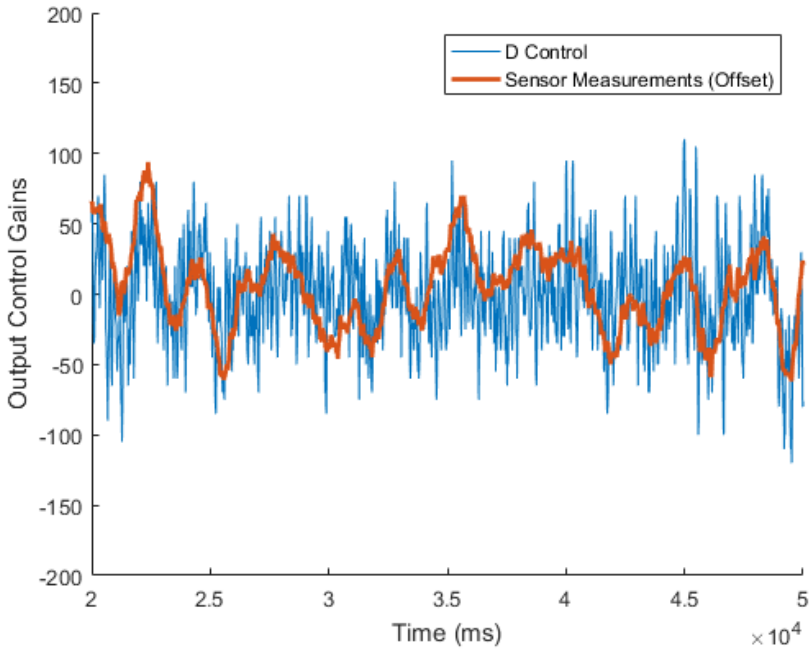


Figure 47: Comparison of derivate control and sensor measurements from experimental testing

There are robust filtering methods that could be introduced to minimize these noise fluctuations in the system. These could be completed by adding more filtering algorithms in software or using hardware filters such as a low-pass filter. Another option is to use higher quality and more accurate onboard sensors, at the expense of increasing costs. Multiple low cost sensors could also be mounted the same direction on a UAV to provide multiple readings. The readings could be combined through basic averaging or more complex methods such as a Kalman Filter to improve the overall accuracy.

It is unclear the exact source of steady state error for the system, as it always maintains an approximate 75mm distance away from the setpoint away from the target surface. It is difficult to solve the problem when such high differential noise is present. It would be recommended in future work to firstly improve the differential error to a minimum before solving the issue of steady state error. Further implementing an integral controller would be prudent if smoothing of

the differential controller is not sufficient.

5.7 Summary

This chapter described the physical system built to functionality test control algorithms. A detailed overview is presented in relation to the hardware, electronics and software used to control the full system. The completed quadrotor was capable of isolating control channels, swapping between autonomous and manual control, logging relevant in flight data and utilizing proximity sensors for control. This was confirmed with a functional test where the system tracked a flat vertical surface using autonomous pitch control. Visually observing the system showed the system worked as expected, although inspecting the logged data more closely showed that there is significant sensor noise that could be reduced to further increase accuracy.

6 Conclusion

6.1 Overview and Contribution Summary

Simulation software was created to model the dynamics of a quadcopter system. The addition of virtual sensors and surface models were implemented on top of this base model to provide a comprehensive simulation system. This completes the stated goals of the thesis and allows for experimentation of various algorithms in relation to proximity sensor control.

Investigation was completed into control algorithms for isolated channels. Considerations that must be made for sensor placement were presented. For flat surfaces, successful control algorithms were implemented for both yaw and pitch control. Algorithms controlling the roll of a UAV could not be completed fully for flat surfaces however, due to the inability to return useful data using only proximity sensors. It is surmised that full autonomous control might be difficult for flat surfaces using only proximity sensors, but by integrating them together with current successful algorithms and hardware there is the potential for a more viable system.

For varied surfaces control potential algorithms are further explored for isolated roll, pitch and yaw channels. The yaw control algorithm showed that it is possible for simple algorithms from basic flat surfaces to be used directly in varied surface cases. A pitch control algorithm is presented for varied surfaces, showing the possibility of look ahead methods which allow for pitch adjustments before encountering an object. Finally a roll control algorithm is explored, that shows how an autonomous system could dynamically slow down and adjust speed as needed. It it postulated that for specific use cases, full autonomous algorithms could be created using only proximity sensors. For a more generic algorithm, it would be recommended to integrate the developed methodology with a proven control methods to further improve potential capability.

Development towards an autonomous testing platform was successful with the creation of a functioning prototype. The system met the goals and requirements of having a fully functional system which is modular, extendable and contains the capacity to isolate and emulate control inputs. Functional testing undertaken validated the system and provided an example experimental test of isolated control algorithms. The result of this was this was successful in flight test, although a further review of flight data shows that there is still work that could be completed into minimizing noise and small control errors.

6.2 Viability for practical use

It has been previously discussed in Chapter 1 that there are many potential options for commercial applications of an autonomous UAV system. After extensive experimentation and research into autonomous control using proximity sensors, it is still believed that this is a valuable area to continue research and development towards.

After consideration of the algorithm exploration in Chapter 4, it is deemed that full autonomous control would remain a difficult task while solely relying on proximity sensors for undefined vertical surfaces. While experimenting on autonomous methods with isolated control channels, it was shown that some of the channels could not be controlled only using these distance measurements, namely the parallel speed through roll control. This is due to the constraints of the limited feedback present, and to overcome this problem alternate sensors and hardware would need to be implemented to supplement the proximity sensor measurements. If current successful control methods such as GPS or computer vision could be integrated together with proximity distance measurements, it is believed that a viable option could be developed for commercial use.

Alternately, it is believed that there are potential options for entirely relying on proximity sensors for specific use cases. If the environment to navigate is known prior, proximity sensors can be set up to allow for the navigation of such areas. For example, if one were to consider a scenario of an indoor inspection bounded on all sides by vertical surfaces, then sensors could be placed in all directions onto the system to detect and fully localize the system on a horizontal plane. This in turn would allow for fully autonomous control while in this constrained environment.

Although not the initial focus of the thesis, there is high potential for further development towards hybrid controller methods. Currently commercial inspections are mostly achieved through manual control, this could be a useful step in the progression towards fully autonomous systems. This viability of a hybrid control system is backed up by the data presented within this thesis, with a working stable prototype of hybrid control quadcopter.

6.3 Further Expansion and Future Work

6.3.1 Simulation Accuracy

Although the implementation of the simulation was successful for the scope of the thesis, there are still further improvements that could be made to improve the overall accuracy of the system. This would allow the simulation more reliably model the real-world and therefore provide better resulting data.

First steps towards this could be simulating all components present on the UAV system more accurately. In the current implementation components such as the proximity sensor or the onboard gyro are modeled perfectly, without any error or noise. This is not an precise model of what occurs in a real-world system, where sensor noise and other disturbances are often present. Development towards this could be completed by integrating more accurate models based on prior research towards all current virtual components.

The simulation presented in this thesis focused on the core dynamics of the system, and opted to neglect all possible disturbances that could be present. In future work the simulation could be extended to include further disturbances that would be valuable for simulation testing. Such examples could include

system drag or external wind disturbances, and these could be implemented extending the functionality of of the simulation further.

6.3.2 Simulation Optimizations

While the initial implementation of the simulation worked effectively, there is still room for further optimizations to improve the overall calculation speed. Within Section 3.7.3, it was identified that the key areas to optimize lay in the current implementation of the virtual sensor beam and determining the intersection with a varied surface. Further optimizations and improvements to this area of the simulation are suggested in improving the local area searches when looking for plane intersections. Improving particular aspect this would significantly improve the overall speed of the simulation, which currently dominates the processing time. Optimizing this and any other areas of the simulator would allow developers more flexibility in modelled surface detail and faster simulation speed to more quickly receive resulting data.

If improved sufficiently enough, there is the possibility of adapting the simulation to run in real-time. This foremostly provides a valuable visual feedback tool for developers and allows to more simply present and explain results to non-technical stakeholders. There could also be the potential for real-time manual control within the simulator, opening pathways for development towards hybrid control capabilities.

6.3.3 Hybrid Control

Throughout this thesis, for initial testing purposes a focus was placed on isolated control inputs. From the developments towards this it was deemed that hybrid control could be a potential area for further exploration with proximity sensors. Physical testing of a prototype system with autonomous control of an isolated channel showed effective tracking results.

Hybrid control for autonomous inspections could be implemented in many different forms to assist operators. One such method could be in maintaining fixed distances close to vertical surfaces. For larger structures it could be possible that an operator must fly a system using only visual line of sight, which when further away is more difficult to accurately determine depth by eye. By removing the need to determine the distance from a vertical surface manually, this allows an operator to focus on the parallel position to the surface. Another option for practical use could be in having onboard safety mechanisms for operators which use proximity sensors to detect when flying to close to obstacles. An autonomous algorithm could then take control of particular channels to move the system to a safe distances from an obstacle or similar. Practical tests in Section 5.6.2 show that swapping between autonomous control and manual control mid-flight was effective and without issue. Such a system could have a commercial application of areas where UAV systems need to be flown within tight indoor spaces.

6.3.4 Autonomous Control for Specific Situations using Proximity Sensors

Developments towards control methods for specific situations using proximity sensors proved highly effective throughout Section 4.4. Furthermore, prior research has been made towards developed autonomous control methods for specific environments with success; further solidifying their viability. Future work could be undertaken into developing autonomous control methods for such specific scenarios, with the benefits of faster development and overall lower cost of the system. These can also have immediate practical uses if the development was focused towards a particular industry problem.

6.3.5 Integration with proven methods

It was surmised that although the methods explored throughout the thesis have potential, integration with proven method are likely to be the most viable option for a robust autonomous system. This comes at the downside of higher complexity and costs around this.

Suggested options include integration with GPS or computer vision. Fusing these methods with proximity sensors helps to overcome their weaknesses. For example lower cost GPS's are traditionally less accurate, which is dangerous when operating near vertical surfaces. If proximity sensors could be used in conjunction with this the GPS could focused on the localization of position in the world while the ranging sensors focus on maintaining a safe or set distance from a surface.

6.3.6 Prototype Development

Although the developed prototype for this thesis was a success, there are still many options to improve it's functionality and reliability.

The current prototype was designed to be a development board, allowing for flexibility in adaptability with the trade-off of the system being heavy and bulky. If a more specialized system was desired to be created, the hardware could easily be reduced to a smaller size. This lower size and weight can be valuable allowing for more mounting options on to many different potential UAVs, both large and small. This also minimize the overall weight, which can be crucial for many airborne systems.

To improve reliability of the system, it would be advised to continue working towards reducing the system noise. There are many options available for this in both hardware and software, discussed in Section 5.6.2.

The completed prototype that was designed can be experimented with further. The limited scope of this thesis did not cover extensive testing, which could easily be explored more comprehensively. It would be advised that further testing be completed for different isolated channels, and following this expansion could be made into testing multiple control channels autonomously at the same time.

Following these steps would allow to slowly expand the functionality of the system while verifying additions as they are integrated. It is clear that the final goal would to have the prototype working with all control channels operated autonomously using only proximity sensors ranging nearby vertical surfaces.

It would also be prudent to test the developed prototype hardware on different UAV platforms. The initial intent of the system is that it would be modular, allowing it to be integrated into a variety of UAV systems. It would be advised to confirm this functionality by testing the hardware across a range of different UAV platforms and ensure correct operation is achieved. It is believed that this should be successful due the radio controller emulation method used, which is common across most UAV systems.

6.4 Closing Remarks

The automation of a UAV system presents a number of challenges to overcome. This is made even more so difficult when adding the limitation of attempting to rely solely on proximity sensors. It was highlighted that although there is some possibly for full autonomous control in specific scenarios, a more likely viable option is to continue exploring ways to integrate this with current proven methods.

Utilizing low-cost proximity sensors for autonomous control was deemed to have potential, although not fully realized in this thesis. Simulations and practical tests for isolated control channels showed promising initial results with successful tracking algorithms developed for a range of surfaces. Developing a system further to rely solely on such sensors or integrating them with current proven methods both have merit towards improving the overall system. This could be in the form of creating a fully autonomous system or even a hybrid control system.

The key goals of this project were achieved, providing a comprehensive knowledge base to build upon in the future. This basis leaves a multitude of options available for future work whether it be towards improving UAV simulations, algorithm development or prototype creation. This knowledge provides a summary of considerations that should be made when developing similar systems, allowing for more informed decision to be made in the future. Detailed development of the simulation, algorithms and prototype are given to show the methodology behind constructing such systems. This allows for solid base to continue exploring any of these areas with the aim of shifting the focus towards new developments and ideas.

7 Bibliography

References

[1] X. xuan Hu, Y. Chen, and H. Luo, "Robust decision making for UAV air-to-ground attack under severe uncertainty," *Journal of Central South University*, vol. 22, no. 11, pp. 4263–4273, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11771-015-2975-y

[2] V. Baiocchi, D. Dominici, and M. Mormile, "UAV application in post-seismic environment," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-1/W2, no. September, pp. 21–25, 2013.

[3] H. Li, B. Wang, L. Liu, G. Tian, T. Zheng, and J. Zhang, "The design and application of SmartCopter: An unmanned helicopter based robot for transmission line inspection," in *Proceedings - 2013 Chinese Automation Congress, CAC 2013*, Changsha, China, 2013, pp. 697–702. [Online]. Available: http://dx.doi.org/10.1109/CAC.2013.6775824

[4] A. Huber, *Micro-Air Vehicles in the Service of Air Force Missions*. Maxwell Air Force Base, Alabama: DIANE Publishing 2002, 2002. [Online]. Available: https://books.google.co.nz/books?id=zEpIbL-A9DQC

[5] J. M. McMichael and M. S. Francis, "Micro Air Vehicles -Toward a New Dimension in Flight," *DARPA Document*, pp. 1–14, 1997. [Online]. Available: http://www.uadrones.net/military/research/1997/0807.htm

[6] P. G. Fahlstrom and T. J. Gleason, "Introduction to UAV Systems: Fourth Edition," *Introduction to UAV Systems: Fourth Edition*, 2012.

[7] L. Meier, "PixHawk Autopilot," 2017. [Online]. Available: https://pixhawk.org/

[8] 3DRobotics, "ArduPilot. Available from http://www.ardupilot.com/," 2015.

[9] Lockheed Martin, "Kestrel Flight Systems & Autopilot," 2017. [Online]. Available: http://www.lockheedmartin.com/us/products/procerus/kestrel-autopilot.html

[10] C. Eschmann, C.-M. Kuo, and C. Boller, "Unmanned Aircraft Systems for Remote Building Inspection and Monitoring," in *Proceedings of the 6th European Workshop on Structural Health Monitoring, July 3-6, 2012, Dresden, Germany*, vol. 2, Dresden, Germany, 2012, pp. 1–8.

[11] I. Sa and P. Corke, "Vertical infrastructure inspection using a Quadcopter and shared autonomy control," in *Springer Tracts in Advanced Robotics*, vol. 92, Matsushima, Miyagi, Japan, 2014, pp. 219–232. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-40686-7_15

[12] K. Mathe and L. Buoniu, "Vision and control for UAVs: A survey of general methods and of inexpensive platforms for infrastructure inspection," *Sensors (Switzerland)*, vol. 15, no. 7, pp. 14 887–14 916, 2015. [Online]. Available: http://dx.doi.org/10.3390/s150714887

[13] J. Yi, L. Zhang, J. Deng, R. Shu, and J. Wang, "GPS/SINS/BARO integrated navigation system for UAV," in *Proceedings - 2010 International Forum on Information Technology and Applications, IFITA 2010*, vol. 3, Kunming, China, 2010, pp. 19–25. [Online]. Available: http://dx.doi.org/10.1109/IFITA.2010.331

[14] P. Campoy, J. F. Correa, I. Mondragón, C. Martínez, M. Olivares, L. Mejías, and J. Artieda, "Computer vision onboard UAVs for civilian tasks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 54, no. 1-3 SPEC. ISS., pp. 105–135, 2009. [Online]. Available: http://dx.doi.org/10.1007/s10846-008-9256-z

[15] G. Morgenthal and N. Hallermann, "Quality Assessment of Unmanned Aerial Vehicle (UAV) Based Visual Inspection of Structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 289–302, 2014. [Online]. Available: http://ase.sagepub.com/lookup/doi/10.1260/1369-4332.17.3.289

[16] N. Gageik, M. Strohmeier, and S. Montenegro, "An autonomous UAV with an optical flow sensor for positioning and navigation," *International Journal of Advanced Robotic Systems*, vol. 10, 2013. [Online]. Available: http://dx.doi.org/10.5772/56813

[17] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–43, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1177352.1177355

[18] K. Çelik, S. J. Chung, M. Clausman, and A. K. Somani, "Monocular vision SLAM for indoor aerial vehicles," pp. 1566–1573, 2009.

[19] Z. X. Yang and S. Jin, "UAV Active SLAM Trajectory Programming Based on Optimal Control," in *Advanced Materials Research*, vol. 765-767, Xiamen, Fujian, China, 2013, pp. 1932–1935. [Online]. Available: http://www.scientific.net/AMR.765-767.1932

[20] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part I," pp. 108–117, 2006.

[21] R. Li, J. Liu, L. Zhang, and Y. Hang, "LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments," in *2014 DGON Inertial Sensors and Systems, ISS 2014 - Proceedings*, Karlsruhe, Germany, 2014, p. IEEE Aerospace and Electronic Systems Society (AES. [Online]. Available: http://dx.doi.org/10.1109/InertialSensors.2014.7049479

[22] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles," 2012.

[23] O. Foundation, "OpenPilot. Available from OpenPilot. Available from http://www.openpilot.org," 2014. [Online]. Available: http://www.openpilot.org

[24] Y. Han and H. Hahn, "Localization and classification of target surfaces using two pairs of ultrasonic sensors," *Robotics and Autonomous Systems*, vol. 33, no. 1, pp. 31–41, 2000.

[25] J. Zhao, Y. Li, D. Hu, and Z. Pei, "Design on altitude control system of quad rotor based on laser radar," *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, pp. 105–109, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7748029/

[26] M. Hebert, "Active and passive range sensing for robotics," pp. 102–110, 2000. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=844046

[27] J. Ryde and N. Hillier, "Performance of laser and radar ranging devices in adverse environmental conditions," *Journal of Field Robotics*, vol. 26, no. 9, pp. 712–727, 2009.

[28] N. Ahmad, R. A. R. Ghazilla, and N. M. Khairi, "Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications," *International Journal of Signal Proccesing Systems*, vol. 1, no. 2, pp. 256–262, 2013. [Online]. Available: http://www.ijsps.com/uploadfile/2013/1128/20131128022014877.pdf

[29] H. Zhao and Z. Wang, "Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended kalman filter for data fusion," *IEEE Sensors Journal*, vol. 12, no. 5, pp. 943–953, 2012.

[30] H. Ferdinando, H. Khoswanto, and D. Purwanto, "Embedded Kalman filter for inertial measurement unit (IMU) on the ATMega8535," *INISTA 2012 - International Symposium on INnovations in Intelligent SysTems and Applications*, 2012.

[31] P. Z. P. Zhang, J. G. J. Gu, E. M. E. Milios, and P. H. P. Huynh, "Navigation with IMU/GPS/digital compass with unscented Kalman filter," *IEEE International Conference Mechatronics and Automation, 2005*, vol. 3, no. July, pp. 1497–1502, 2005.

[32] P. Van Turennout, G. Honderd, and L. J. Van Schelven, "Wall-following control of a mobile robot," *Proc IEEE International Conference on Robotics and Automation ICRA1992*, pp. 280–285, 1992.

[33] G. Nugraha, R. A. Haris, A. W. Multazam, K. Mutijarsa, and W. Adiprawita, "Design and implementation of Active Object Computing Model for a wall following mobile robot," *2014 IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, pp. 258–262, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/7295898/

[34] A. Bemporad, M. D. Marco, and A. Tesi, "Wall-following controllers for sonar-based mobile robots," *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 3, no. December, pp. 3063–3068, 1997.

[35] Y. Liu, R. Fu, J. Wang, Y. Ou, X. Wu, and A. Peng, "A wall-following strategy for mobile robots based on self-convergence," pp. 824–828, 2011.

[36] A. Imhof, M. Oetiker, and B. Jensen, "Wall following for autonomous robot navigation," pp. 1–4, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6473370

[37] Y. Ando and S. Yuta, "Following a wall by an autonomous mobile robot with a sonar-ring," *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2599–2606, 1995.

[38] S. Fazli and L. Kleeman, "Wall following and obstacle avoidance results from a multi-DSP sonar ring on a mobile robot," *IEEE International Conference Mechatronics and Automation, 2005*, vol. 1, no. July, pp. 432–437, 2005.

[39] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to MAV navigation," pp. 3923–3929, 2013.

[40] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, "A UAV system for inspection of industrial facilities," in *IEEE Aerospace Conference Proceedings*, Big Sky, MT, United states, 2013. [Online]. Available: http://dx.doi.org/10.1109/AERO.2013.6496959

[41] T. Bresciani, "Modelling , Identification and Control of a Quadrotor Helicopter," *English*, vol. 4, no. October, p. 213, 2008. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Modelling+,+Identification+and+C

[42] H. C. T. E. Fernando, A. T. A. De Silva, M. D. C. De Zoysa, K. A. D. C. Dilshan, and S. R. Munasinghe, "Modelling, simulation and implementation of a quadrotor UAV," *2013 IEEE 8th International Conference on Industrial and Information Systems, ICIIS 2013 - Conference Proceedings*, pp. 207–212, 2013.

[43] K. Patel and J. Barve, "Modeling, simulation and control study for the quad-copter UAV," *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, pp. 1–6, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7036590

[44] A. R. Patel, M. a. Patel, and D. R. Vyas, "Modeling and analysis of quadrotor using sliding mode control," pp. 111–114, 2012.

[45] A. Nemati, M. Sarim, M. Hashemi, and M. Kumar, "Autonomous Wall-Following Based Navigation of Unmanned Aerial Vehicles in Indoor Environments," in *AIAA Infotech @ Aerospace*, no. JANUARY, Kissimmee, FL, United states, 2015, p. 8. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/6.2015-0989

[46] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[47] P. Pounds, R. Mahony, and P. Corke, "Modelling and Control of a Quad-Rotor Robot," *East*, vol. 4, p. 44, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.1200&rep=rep1&type=pdf

[48] R. Vepa, "Flight Dynamics, Simulation, and Control For Rigid and Flexible Aircraft," 2014.

[49] A. Sharifi and H. Nobahari, "Multiple model filters applied to wind model estimation for a fixed wing UAV," *2016 7th International Conference on Mechanical and Aerospace Engineering (ICMAE)*, pp. 109–115, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7549518/

[50] W. Kaidi, L. Chuntao, C. Peng, and F. Ying, "Design of real-time and multi-task UAV simulation system based on rapid prototyping," pp. 930–936, 2016.

[51] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7628 LNAI, pp. 400–411, 2012.

[52] "Gazebo," 2014. [Online]. Available: http://gazebosim.org/

[53] F. Wang, J.-Q. Cui, B.-M. Chen, and T. H. Lee, "A Comprehensive UAV Indoor Navigation System Based on Vision Optical Flow and Laser FastSLAM," *Acta Automatica Sinica*, vol. 39, no. 11, pp. 1889–1899, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874102913600804

[54] J. F. Roberts, T. S. Stirling, J.-C. Zufferey, and D. Floreano, "Quadrotor Using Minimal Sensing For Autonomous Indoor Flight," *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, no. September, pp. 17–21, 2007.

[55] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[56] MathWorks, "MATLAB Key Features," 2017. [Online]. Available: https://au.mathworks.com/products/matlab/features.html

[57] A. Rodić and G. Mester, "The modeling and simulation of an autonomous quad-rotor microcopter in a virtual outdoor scenario," pp. 107–122, 2011.

[58] M. D. Schmidt, "Simulation and Control of a Quadrotor Unmanned Aerial Vehicle," Ph.D. dissertation, University of Kentucky, 2011.

[59] A. Gibiansky, "Quadcopter Dynamics , Simulation , and Control Introduction Quadcopter Dynamics," pp. 1–18, 2012. [Online]. Available: http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/

[60] R. Tan and M. Kumar, "Proportional Navigation (PN) Based Tracking of Ground Targets by Quadrotor UAVs," p. V001T01A004, 2013. [Online]. Available: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?doi=10.1115/DSCC2013-3887

[61] A. H. Ahmed, A. N. Ouda, A. M. Kamel, and Y. Z. Elhalwagy, "Attitude stabilization and altitude control of Quadrotor," *2016 12th International Computer Engineering Conference, ICENCO 2016: Boundless Smart Societies*, pp. 123–130, 2017.

[62] A. Zulu and S. John, "A Review of Control Algorithms for Autonomous Quadrotors," *Open Journal of Applied Sciences*, vol. 04, no. 14, pp. 547–556, 2014. [Online]. Available: http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/ojapps.2014.414053

[63] Z. Yao, X. Bin, Y. Qiang, L. Yang, and W. Fu, "Autonomous control system for the quadrotor unmanned aerial vehicle," pp. 4862–4867, 2012.

[64] M. Ishihara, M. Shiina, and S.-n. Suzuki, "Evaluation of Method of Measuring Distance Between Object and Walls Using Ultrasonic Sensors," *Journal of Asian Electric Vehicles*, vol. 7, no. 1, pp. 1207–1211, 2009. [Online]. Available: http://joi.jlc.jst.go.jp/JST.JSTAGE/jaev/7.1207?from=CrossRef

[65] Y. Okubo, C. Ye, and J. Borenstein, "Characterization of the Hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation," pp. 733 212:1–733 212:10, 2009.

[66] A. Bachrach, A. De Winter, R. He, G. Hemann, S. Prentice, and N. Roy, "RANGE - Robust autonomous navigation in GPS-denied environments," pp. 1096–1097, 2010.

[67] M. Tailanian, S. Paternain, R. Rosa, and R. Canetti, "Design and implementation of sensor data fusion for an autonomous quadrotor," pp. 1431–1436, 2014.

[68] W. Zheng, J. Wang, and Z. Wang, "Multi-sensor fusion based real-time hovering for a quadrotor without GPS in assigned position," pp. 3605–3610, 2016.

[69] A. Zul Azfar and D. Hazry, "A simple approach on implementing IMU sensor fusion in PID controller for stabilizing quadrotor flight control," *Proceedings - 2011 IEEE 7th International Colloquium on Signal Processing and Its Applications, CSPA 2011*, pp. 28–32, 2011.

[70] A. Chan, S. Tan, and C. Kwek, "Sensor data fusion for attitude stabilization in a low cost Quadrotor system," *Consumer Electronics (ISCE), ...*, pp. 34–39, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5973778

[71] Arduino AG, "Arduino MEGA 2560," 2017. [Online]. Available: http://www.arduino.org/products/boards/arduino-mega-2560

[72] C. C. Zhih, S. K. V. Ragavan, and M. Shanmugavel, "Development of a simple, low-cost autopilot system for multi-rotor UAVs," pp. 285–289, 2016.

[73] D. J. Esteves, A. Moutinho, and J. R. Azinheira, "Stabilization and altitude control of an indoor low-cost quadrotor: Design and experimental results," pp. 150–155, 2015.

[74] J. Toji, Y. Iwata, and H. Ichihara, "Building quadrotors with Arduino for indoor environments," *2015 10th Asian Control Conference: Emerging Control Techniques for a Sustainable World, ASCC 2015*, 2015.

[75] M. Shahrieel, M. Aras, K. Mazlan, A. Ahmad, N. M. Ali, and M. Safirin, "Analysis Performances of Laser Range Finder and Blue LED for Autonomous Underwater Vehicle Analysis Performances of Laser Range Finder and Blue LED for Autonomous Underwater Vehicle (AUV)," no. January 2017, pp. 171–176, 2016.

[76] H. Qin, Y. Bi, K. Z. Ang, K. Wang, J. Li, M. Lan, M. Shan, and F. Lin, "A stereo and rotating laser framework for UAV navigation in GPS denied environment," pp. 6061–6066, 2016.

[77] F. Wang, J. Cui, S. K. Phang, B. M. Chen, and T. H. Lee, "A mono-camera and scanning laser range finder based UAV indoor navigation system," *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, pp. 694–701, 2013.

[78] Garmin Ltd, "LIDAR-Lite v2 "Blue Label"," 2017. [Online]. Available: https://www.pulsedlight3d.com/products/lidar-lite-v2-blue-label.html

[79] H. Wu, W. Su, and Z. Liu, "PID controllers: Design and tuning methods," pp. 808–813, 2014.

8 Appendices

Appendix A: Simulation Profile

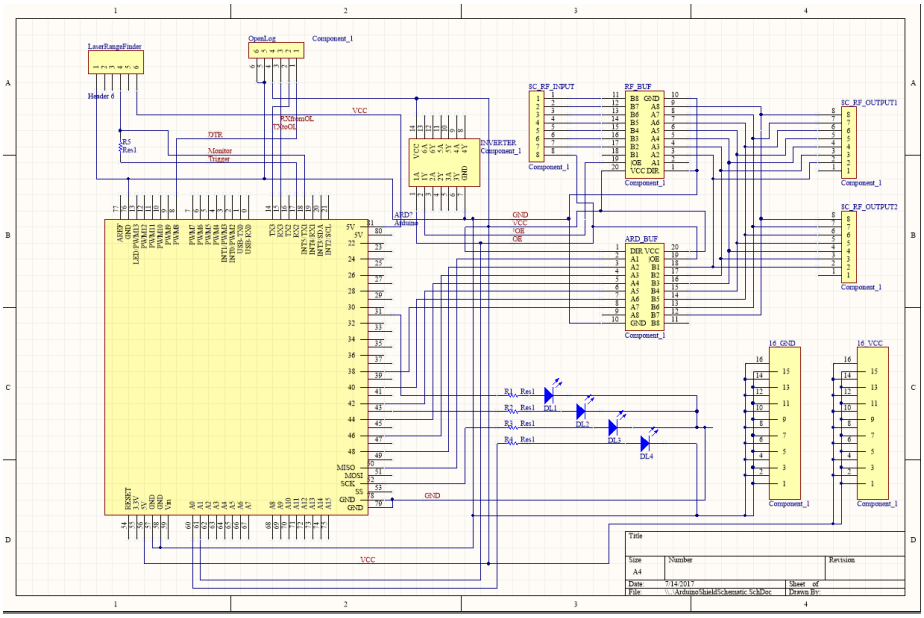
Quadcopter Simulation Profile Summary

Generated 28-Jul-2017 17:01:58 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
simulation_performance	1	11.309 s	0.215 s	
..._full_intersection_search_distance_v2	632	10.086 s	2.672 s	
dot	879842	5.895 s	5.895 s	
plane_line_intersect	220460	3.871 s	0.907 s	
cross	877840	1.194 s	1.194 s	
...egion_intersection_search_distance_v2	1000	0.812 s	0.257 s	
quatrotate	3000	0.105 s	0.022 s	
quat2dcm	3000	0.083 s	0.044 s	
quatnormalize	3000	0.039 s	0.014 s	
sensor_plane_intersection_distance	1000	0.036 s	0.010 s	
quatmod	3000	0.024 s	0.024 s	
angle2quat	1000	0.020 s	0.020 s	
Rotation_BF_to_EF	1000	0.011 s	0.011 s	
Angular_Velocity_BF	1000	0.009 s	0.009 s	
Desired_Rotor_Speeds	1000	0.009 s	0.009 s	
Angular_Accleration_Euler	1000	0.004 s	0.004 s	
generate_flat_plane	1	0.002 s	0.002 s	

Self time is the time spent in a function excluding the time spent in its child functions.
Self time also includes overhead resulting from the process of profiling.

Appendix B: Schematic Design



Appendix C: PCB Design